Product or cutomer segmentation on Classplus 1 year data

In [2]:
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

In [3]:
```
df1 = pd.read_excel(r"E:\Class Plus Case Study\PA_ASSIGNMENT.xlsx", sheet_name= "Tra
df2 = pd.read_excel(r"E:\Class Plus Case Study\PA_ASSIGNMENT.xlsx", sheet_name="GMV"
```

In [329…
```
df1
```

Out[329…

	OrgId	Jun'19	Jul'19	Aug'19	Sep'19	Oct'19	Nov'19	Dec'19	Jan'20	Feb'20	Mar'20	Apr'
0	C1	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	(
1	C2	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	4
2	C3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7
3	C4	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	3.0	2.0	1
4	C5	0.0	1.0	1.0	0.0	0.0	7.0	0.0	0.0	1.0	0.0	(
…	…	…	…	…	…	…	…	…	…	…	…	
229	C230	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	16.0	1
230	C231	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	2.0	(
231	C232	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	2
232	C233	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	8(
233	C234	0.0	0.0	0.0	1.0	0.0	0.0	4.0	1.0	7.0	16.0	1

234 rows × 13 columns

In [5]:
```
df2
```

Out[5]:

	OrgId	Jun'19	Jul'19	Aug'19	Sep'19	Oct'19	Nov'19	Dec'19	Jan'20	
0	C1	99.999998	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	C2	NaN	NaN	NaN	NaN	NaN	NaN	2979.000019	NaN	
2	C3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	C4	NaN	NaN	NaN	NaN	NaN	NaN	300.000005	NaN	1(
4	C5	NaN	1.0	999.999956	NaN	NaN	7000.000134	NaN	NaN	
…	…	…	…	…	…	…	…	…	…	
229	C230	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

	OrgId	Jun'19	Jul'19	Aug'19	Sep'19	Oct'19	Nov'19	Dec'19	Jan'20
230	C231	NaN	NaN	NaN	NaN	NaN	NaN	1200.000029	NaN
231	C232	NaN	NaN	NaN	NaN	NaN	NaN	99.999999	NaN
232	C233	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
233	C234	NaN	NaN	NaN	1.0	NaN	NaN	402.000011	399.000011

234 rows × 13 columns

In [6]:
```
df1  = df1.fillna(0)
df2 = df2.fillna(0)
```

In [7]:
```
df2.describe()
```

Out[7]:

	Jun'19	Jul'19	Aug'19	Sep'19	Oct'19	Nov'19	
count	234.000000	2.340000e+02	234.000000	2.340000e+02	2.340000e+02	234.000000	2.3
mean	2427.360278	1.009258e+04	13524.957187	1.916877e+04	1.681934e+04	12715.312491	1.5
std	25542.242842	8.692593e+04	86152.452484	1.527861e+05	9.384452e+04	68313.986485	8.5
min	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.0
25%	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.0
50%	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.0
75%	0.000000	0.000000e+00	0.000000	1.075000e+01	1.000000e+02	420.249993	5.9
max	381499.992477	1.076371e+06	865057.010134	2.218779e+06	1.070517e+06	731501.575584	1.0

Basic Stats for Transaction sheet

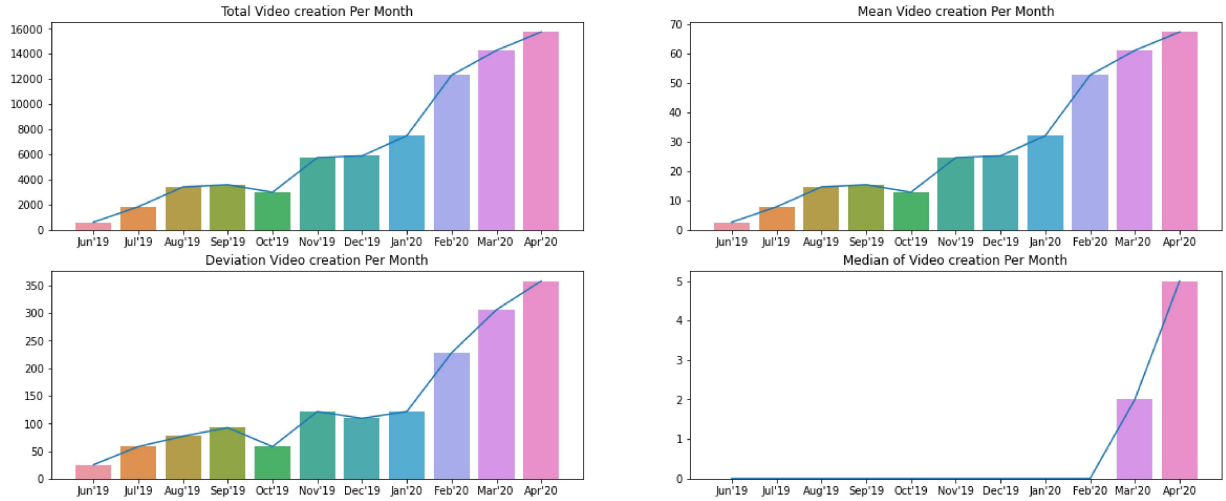
In [8]:
```
df1.describe()
```

Out[8]:

	Jun'19	Jul'19	Aug'19	Sep'19	Oct'19	Nov'19	Dec'19	Ja
count	234.000000	234.00000	234.000000	234.000000	234.000000	234.000000	234.000000	234.00
mean	2.508547	7.75641	14.512821	15.277778	12.713675	24.504274	25.089744	31.97
std	25.049692	57.90080	76.689958	92.255452	57.949653	121.353888	109.229355	121.48
min	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	0.000000	0.00000	0.000000	1.000000	2.000000	2.750000	3.000000	5.00
max	376.000000	824.00000	794.000000	1286.000000	691.000000	1474.000000	1208.000000	982.00

In [9]:
```
#plt.figure(figsize= (10,10))

fig, axis = plt.subplots( nrows= 2,ncols=2, figsize=(20, 8))
```

```
colms = df1.columns[1:-1]

sns.barplot(x = colms, y= [df1[i].sum() for i in colms], ax = axis[0,0]).set(title='
sns.lineplot(x = colms, y= [df1[i].sum() for i in colms], ax = axis[0,0]).set(title=

sns.barplot(x = colms, y= [df1[i].mean() for i in colms], ax = axis[0,1]).set(title=
sns.lineplot(x = colms, y= [df1[i].mean() for i in colms], ax = axis[0,1]).set(title

sns.barplot(x = colms, y= [df1[i].std() for i in colms], ax = axis[1,0]).set(title='
sns.lineplot(x = colms, y= [df1[i].std() for i in colms], ax = axis[1,0]).set(title=

sns.barplot(x = colms, y= [df1[i].median() for i in colms], ax = axis[1,1]).set(titl
sns.lineplot(x = colms, y= [df1[i].median() for i in colms], ax = axis[1,1]).set(tit
plt.show()
```



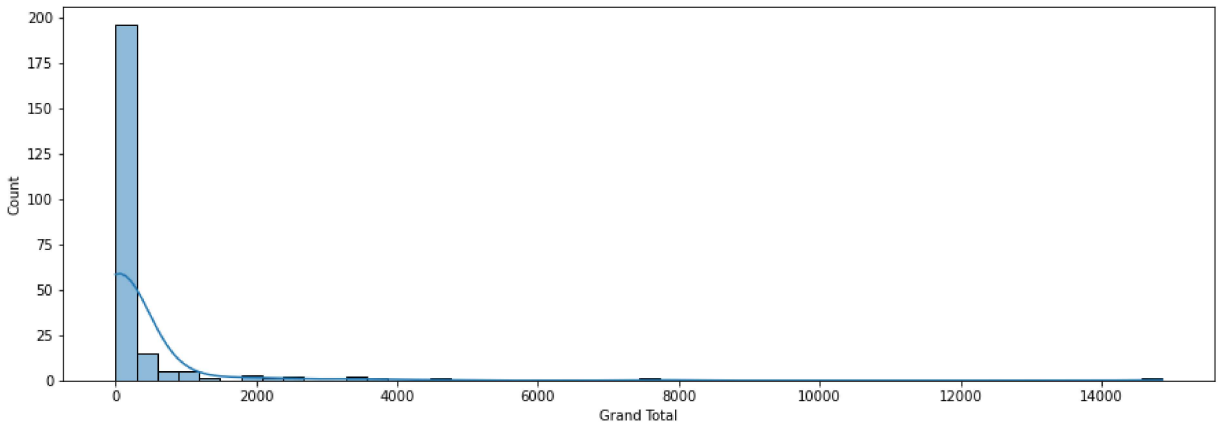
Observation:

1. From total and mean video creation plot we can say with time the number video creation is increasing Which indicates this platform is doing good for the creators

2. Now from deviation plot since data deviation also increasing it could possibly because of those who are keep increasing their video production capcity than others. And they are more serious about it.

3. And from median video creation plot one intresting thing we notice that most of the creators like more than 75% (from above description table) were not even creating videos before feb20 or they were not even present in the platform.

In [10]:
```
plt.plot = plt.figure(figsize= (15,5))
sns.histplot(x = df1['Grand Total'], bins = 50, kde = True)
plt.show()
```



In [11]:
```
for i in range(0,101,10):
    v = np.percentile(df1['Grand Total'],i)
    print("{}'th Percentile value of grand total: {}".format(i,v))
```

```
0'th Percentile value of grand total: 3.0
10'th Percentile value of grand total: 3.0
20'th Percentile value of grand total: 4.0
30'th Percentile value of grand total: 7.0
40'th Percentile value of grand total: 12.0
50'th Percentile value of grand total: 17.0
60'th Percentile value of grand total: 31.0
70'th Percentile value of grand total: 52.19999999999999
80'th Percentile value of grand total: 162.4
90'th Percentile value of grand total: 579.9000000000003
100'th Percentile value of grand total: 14865.0
```

In [12]:
```
for i in range(90,101,1):
    v = np.percentile(df1['Grand Total'],i)
    print("{}'th Percentile value of grand total: {}".format(i,v))
```

```
90'th Percentile value of grand total: 579.9000000000003
91'th Percentile value of grand total: 636.87
92'th Percentile value of grand total: 771.5600000000006
93'th Percentile value of grand total: 1032.9199999999998
94'th Percentile value of grand total: 1120.3599999999997
95'th Percentile value of grand total: 1505.2999999999975
96'th Percentile value of grand total: 1979.8799999999987
97'th Percentile value of grand total: 2499.2799999999997
98'th Percentile value of grand total: 3346.9800000000005
99'th Percentile value of grand total: 4211.569999999991
100'th Percentile value of grand total: 14865.0
```

Clearly only a small person of people producing really high number of contents

Lets see if it has a tendency of following 80-20 principle or power law distrbution

In [14]:
```
df1_sorted = df1.sort_values('Grand Total')
```

In [15]:
```
p_20 = df1_sorted['Grand Total'][-int(234*.2):].sum()/df1['Grand Total'].sum()
p_10 = df1_sorted['Grand Total'][-int(234*.1):].sum()/df1['Grand Total'].sum()

print(" Total percentage of videos produced by top 20% of people - ", p_20)
print(" Total percentage of videos produced by top 10% of people - ", p_10)
```

```
Total percentage of videos produced by top 20% of people -  0.935275695169403
Total percentage of videos produced by top 10% of people -  0.819805040740791
```

In [16]:
```
# Let's see the numbers monthwise for top 10% of people
df1_sorted[colms][-int(234*.1):].sum()/df1[colms].sum()
```

Out[16]:
```
Jun'19     0.778535
```

```
Jul'19     0.940496
Aug'19     0.901060
Sep'19     0.832727
Oct'19     0.766387
Nov'19     0.884025
Dec'19     0.880770
Jan'20     0.810988
Feb'20     0.879561
Mar'20     0.791159
Apr'20     0.734409
dtype: float64
```

So I would like to state that the dataset has a tendecy of following 80-20 principle, here instead 20, top 10 percent people are producing almost 80 percent of the videos

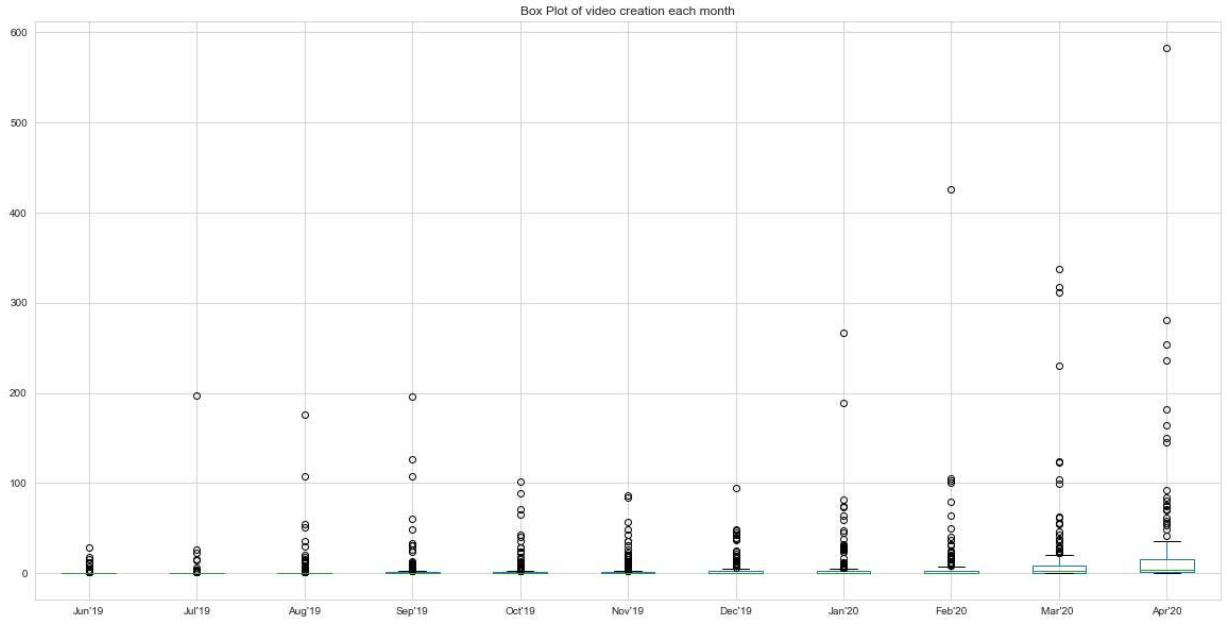
In [17]:
```
df1_r = df1_sorted[df1_sorted['Grand Total'] < np.percentile(df1_sorted['Grand Total
df1_r['Grand Total'][-int(len(df1_r)*.2):].sum()/df1_r['Grand Total'].sum()
```

Out[17]:
```
0.8057197855080435
```

In [18]:
```
df1_r[colms][-int(len(df1_r)*.2):].sum()/df1_r[colms].sum()
```

Out[18]:
```
Jun'19     0.613636
Jul'19     0.878049
Aug'19     0.875606
Sep'19     0.826859
Oct'19     0.770115
Nov'19     0.776675
Dec'19     0.850197
Jan'20     0.907258
Feb'20     0.851827
Mar'20     0.792277
Apr'20     0.752785
dtype: float64
```

In [353…
```
plt.figure(figsize = (20,10))
df1_r[colms].boxplot(grid=True, fontsize=10)
plt.title("Box Plot of video creation each month")
plt.show()
```



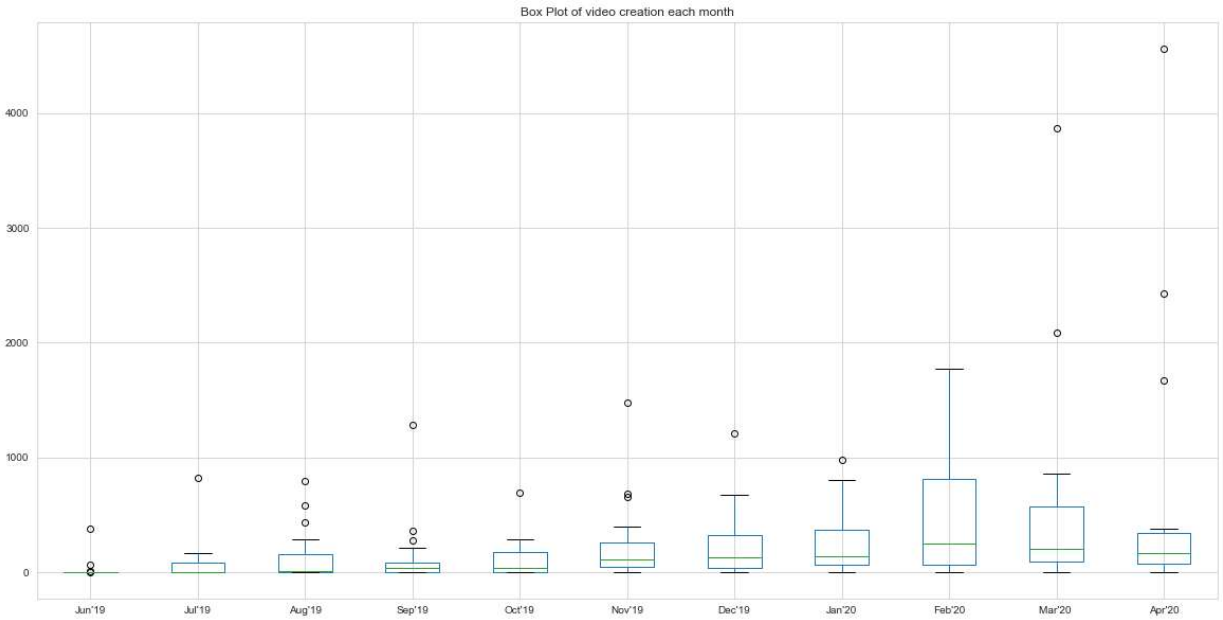
Now here I have selected creators with total less than arround 636 but still some month has total videos close to 600 which I want consider as outliers and can inspect them later

In [20]:
```
# Ditribution among top 9%
df1_lr = df1_sorted[df1_sorted['Grand Total'] >= np.percentile(df1_sorted['Grand Tot
```

```
df1_lr['Grand Total'][-int(len(df1_lr)*.2):].sum()/df1_lr['Grand Total'].sum()
```

Out[20]: 0.5178930302467364

In [368…
```
plt.figure(figsize = (20,10))
df1_lr[colms].boxplot(grid=True, fontsize=10)
plt.title("Box Plot of video creation each month")
plt.show()
```



In [22]:
```
top_10 = df1_lr.OrgId
```

In [23]:
```
len(top_10)
```

Out[23]: 21

Conclusion:

1. These top 9 percentile creator seems to be here for a longer period of time and with time they seems to be increasing there contents, also the number of contents indicates they are not an individual but studios with a good capacity video production. Since they are taking interest for a long period of time that means Classplus platform really working good for them. Which is a good sign

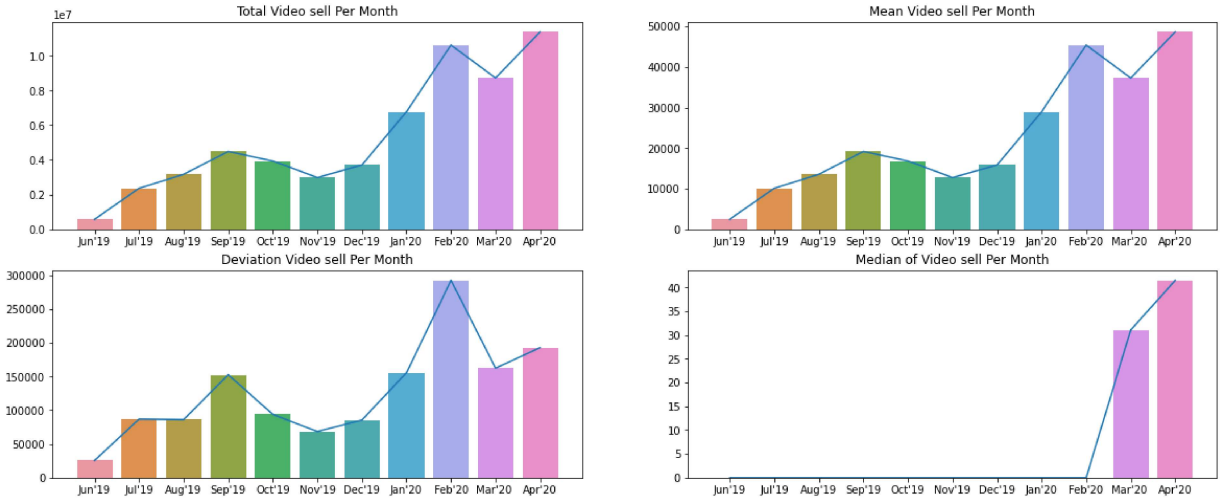
1. Also they produce almost 80% of total videos, so they should be taken care of.

Basic Stats for GMV sheet

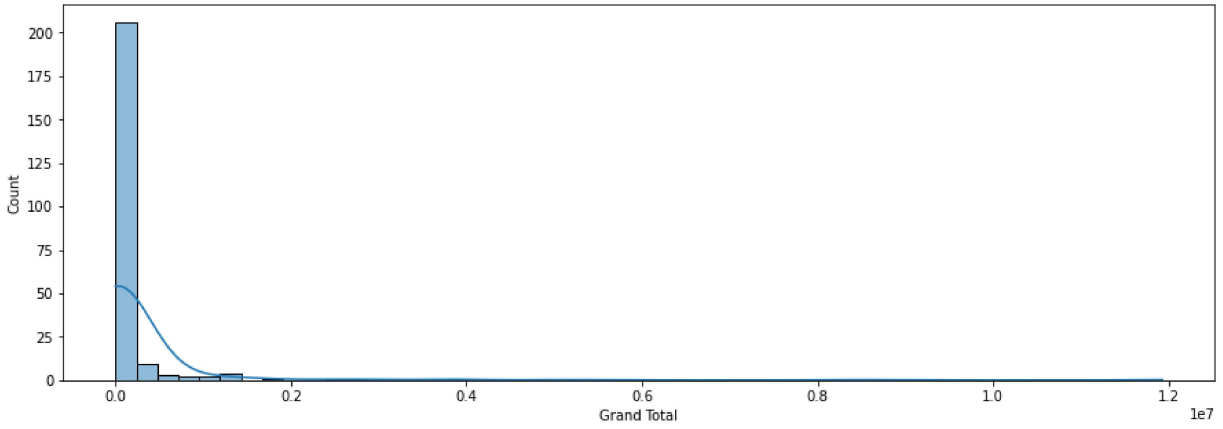
In [24]:
```
df2.columns = df1.columns
```

In [25]:
```
#plt.figure(figsize= (10,10))

fig, axis = plt.subplots( nrows= 2,ncols=2, figsize=(20, 8))


colms2 = df2.columns[1:-1]

sns.barplot(x = colms, y= [df2[i].sum() for i in colms], ax = axis[0,0]).set(title='
sns.lineplot(x = colms, y= [df2[i].sum() for i in colms], ax = axis[0,0]).set(title=
```

```
sns.barplot(x = colms, y= [df2[i].mean() for i in colms], ax = axis[0,1]).set(title=
sns.lineplot(x = colms, y= [df2[i].mean() for i in colms], ax = axis[0,1]).set(title

sns.barplot(x = colms, y= [df2[i].std() for i in colms], ax = axis[1,0]).set(title='
sns.lineplot(x = colms, y= [df2[i].std() for i in colms], ax = axis[1,0]).set(title=

sns.barplot(x = colms, y= [df2[i].median() for i in colms], ax = axis[1,1]).set(titl
sns.lineplot(x = colms, y= [df2[i].median() for i in colms], ax = axis[1,1]).set(tit
plt.show()
```



In [26]:
```
plt.plot = plt.figure(figsize= (15,5))
sns.histplot(x = df2['Grand Total'], bins = 50, kde = True)
plt.show()
```



In [27]:
```
for i in range(0,101,10):
    v = np.percentile(df2['Grand Total'],i)
    print("{}'th Percentile value of grand total: {}".format(i,v))
```

```
0'th Percentile value of grand total: 2.999999916180962
10'th Percentile value of grand total: 12.002999746799457
20'th Percentile value of grand total: 51.637999422848125
30'th Percentile value of grand total: 253.61800691336353
40'th Percentile value of grand total: 711.2040066346517
50'th Percentile value of grand total: 2121.999988436688
60'th Percentile value of grand total: 6911.2000892639035
70'th Percentile value of grand total: 19756.71817368415
80'th Percentile value of grand total: 68635.0288904316
90'th Percentile value of grand total: 328164.5239552579
100'th Percentile value of grand total: 11930639.626307186
```

In [28]:
```
for i in range(90,101,1):
    v = np.percentile(df2['Grand Total'],i)
    print("{}'th Percentile value of grand total: {}".format(i,v))
```

```
90'th Percentile value of grand total: 328164.5239552579
```

```
91'th Percentile value of grand total: 376598.4625669058
92'th Percentile value of grand total: 489108.90432557894
93'th Percentile value of grand total: 677559.9711930151
94'th Percentile value of grand total: 893677.3623285928
95'th Percentile value of grand total: 1188801.1180923218
96'th Percentile value of grand total: 1322248.60126882
97'th Percentile value of grand total: 1841731.9913068356
98'th Percentile value of grand total: 3047157.4237821028
99'th Percentile value of grand total: 4755291.727028895
100'th Percentile value of grand total: 11930639.626307186
```

In [29]:
```
df2_sorted = df2.sort_values('Grand Total')
```

In [30]:
```
p_20 = df2_sorted['Grand Total'][-int(234*.2):].sum()/df2['Grand Total'].sum()
p_10 = df2_sorted['Grand Total'][-int(234*.1):].sum()/df2['Grand Total'].sum()

print(" Total percentage of videos produced by top 20% of people - ", p_20)
print(" Total percentage of videos produced by top 10% of people - ", p_10)
```

```
Total percentage of videos produced by top 20% of people -  0.975341069913418
Total percentage of videos produced by top 10% of people -  0.9033305604772583
```

In [31]:
```
# Let's see the numbers monthwise for top 10% of people
df2_sorted[colms][-int(234*.2):].sum()/df2[colms].sum()
```

Out[31]:
```
Jun'19     0.967591
Jul'19     0.989880
Aug'19     0.961396
Sep'19     0.979400
Oct'19     0.977887
Nov'19     0.958994
Dec'19     0.975878
Jan'20     0.985993
Feb'20     0.987964
Mar'20     0.972120
Apr'20     0.962556
dtype: float64
```

In [32]:
```
df2_lr = df2_sorted[df2_sorted['Grand Total'] >= np.percentile(df2_sorted['Grand Tot
top_10_soldout = df2_lr.OrgId
```

Overlap between top 20 creator and top 20 percentile GMV

In [33]:
```
len(set.intersection(set(top_10),set(top_10_soldout)))
```

Out[33]:
```
12
```

In [34]:
```
set(top_10) - set(top_10_soldout)
```

Out[34]:
```
{'C121', 'C149', 'C171', 'C195', 'C206', 'C207', 'C223', 'C41', 'C69'}
```

In [35]:
```
set(top_10_soldout) - set(top_10)
```

Out[35]:
```
{'C100', 'C125', 'C128', 'C15', 'C182', 'C204', 'C215', 'C51', 'C9'}
```

In [37]:
```
#No of people in this range of top 20 percenile value
len(top_20),len(top_20_soldout)
```

Out[37]:
```
(47, 47)
```

In [370…
```
# Creator presnt in either of datsets top 20 percentiles
top_20 = df1_sorted[df1_sorted['Grand Total'] >= np.percentile(df1_sorted['Grand Tot
```

```
top_20_soldout = df2_sorted[df2_sorted['Grand Total'] >= np.percentile(df2_sorted['G
len(set.union(set(top_20),set(top_20_soldout)))
```

Out[370… 59

In [371…
```
# Creator presnt in both datasets top 20 percentiles
top_20 = df1_sorted[df1_sorted['Grand Total'] >= np.percentile(df1_sorted['Grand Tot
top_20_soldout = df2_sorted[df2_sorted['Grand Total'] >= np.percentile(df2_sorted['G
len(set.intersection(set(top_20),set(top_20_soldout)))
```

Out[371… 35

In [38]:
```
# There is people who creators who creats high number of videos but not in the range
len(set(top_20) - set(top_20_soldout))
```

Out[38]: 12

In [369…
```
print('Top Content creators with very low engagement contents :',set(top_20) - set(t
```

```
Top Content creators with very low engagement contents : {'C121', 'C69', 'C171', 'C5
4', 'C147', 'C89', 'C172', 'C225', 'C149', 'C224', 'C214', 'C199'}
```

These 12 people should review their contents since they are not in the that range where most of the videos are sold.

But They are in the range of most consistent one and if they are still producing gaining profit, then it indicates that they have high price of selling

Check if there is a diffence between months of producing and selling

Transaction and GMV datasets are replicating each others, also timing of buying and selling of thosem I doubt that they are live classes or contents with limited access

In [40]:
```
def exists(x):
    if x !=0: return 1
    return 0
```

In [41]:
```
non_live = {}
for i in colms:
    non_live[i] = np.sum(df1[i].apply(exists) - df2[i].apply(exists)) #Each column e
print(" Print number non_live classes monthwise \n", non_live)
```

```
 Print number non_live classes monthwise
 {"Jun'19": 0, "Jul'19": 0, "Aug'19": 0, "Sep'19": 0, "Oct'19": 0, "Nov'19": 0, "De
c'19": 0, "Jan'20": 0, "Feb'20": 0, "Mar'20 ": 0, "Apr'20": 0}
```

Seems like my assumption is true that all these contents are live or have a expiry date of access

Watch per video Table

In [74]:
```
df_frqnc = df2.iloc[:,1:]/df1.iloc[:,1:]
df_frqnc['OrgId'] = df1.OrgId
df_frqnc.fillna(0)
```

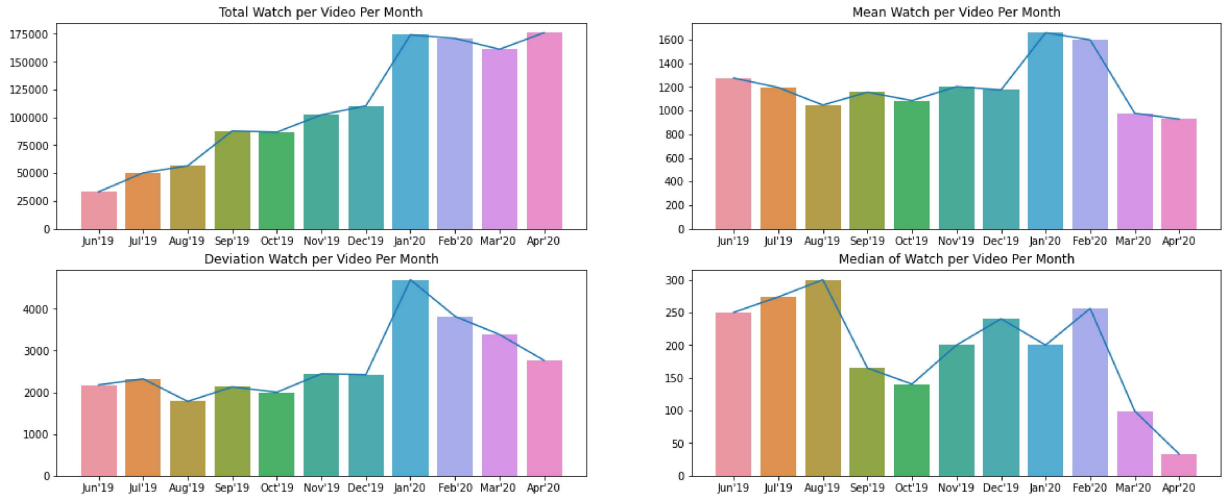
Out[74]:

	Jun'19	Jul'19	Aug'19	Sep'19	Oct'19	Nov'19	Dec'19	Jan'20	Feb'
0	49.999999	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.0000
1	0.000000	0.0	0.000000	0.0	0.0	0.000000	993.000006	0.000000	0.0000

	Jun'19	Jul'19	Aug'19	Sep'19	Oct'19	Nov'19	Dec'19	Jan'20	Feb'
2	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.0000
3	0.000000	0.0	0.000000	0.0	0.0	0.000000	300.000005	0.000000	3573.3332
4	0.000000	1.0	999.999956	0.0	0.0	1000.000019	0.000000	0.000000	1.0000
...	...	...	...	...	...	...	...	...	
229	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.0000
230	0.000000	0.0	0.000000	0.0	0.0	0.000000	1200.000029	0.000000	0.0000
231	0.000000	0.0	0.000000	0.0	0.0	0.000000	49.999999	0.000000	0.0000
232	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.0000
233	0.000000	0.0	0.000000	1.0	0.0	0.000000	100.500003	399.000011	191.8571

234 rows × 13 columns

In [43]:
```
#plt.figure(figsize= (10,10))

fig, axis = plt.subplots( nrows= 2,ncols=2, figsize=(20, 8))


#colms2 = df2.columns[1:-1]

sns.barplot(x = colms, y= [df_frqnc[i].sum() for i in colms], ax = axis[0,0]).set(ti
sns.lineplot(x = colms, y= [df_frqnc[i].sum() for i in colms], ax = axis[0,0]).set(t

sns.barplot(x = colms, y= [df_frqnc[i].mean() for i in colms], ax = axis[0,1]).set(t
sns.lineplot(x = colms, y= [df_frqnc[i].mean() for i in colms], ax = axis[0,1]).set(

sns.barplot(x = colms, y= [df_frqnc[i].std() for i in colms], ax = axis[1,0]).set(ti
sns.lineplot(x = colms, y= [df_frqnc[i].std() for i in colms], ax = axis[1,0]).set(t

sns.barplot(x = colms, y= [df_frqnc[i].median() for i in colms], ax = axis[1,1]).set
sns.lineplot(x = colms, y= [df_frqnc[i].median() for i in colms], ax = axis[1,1]).se
plt.show()
```

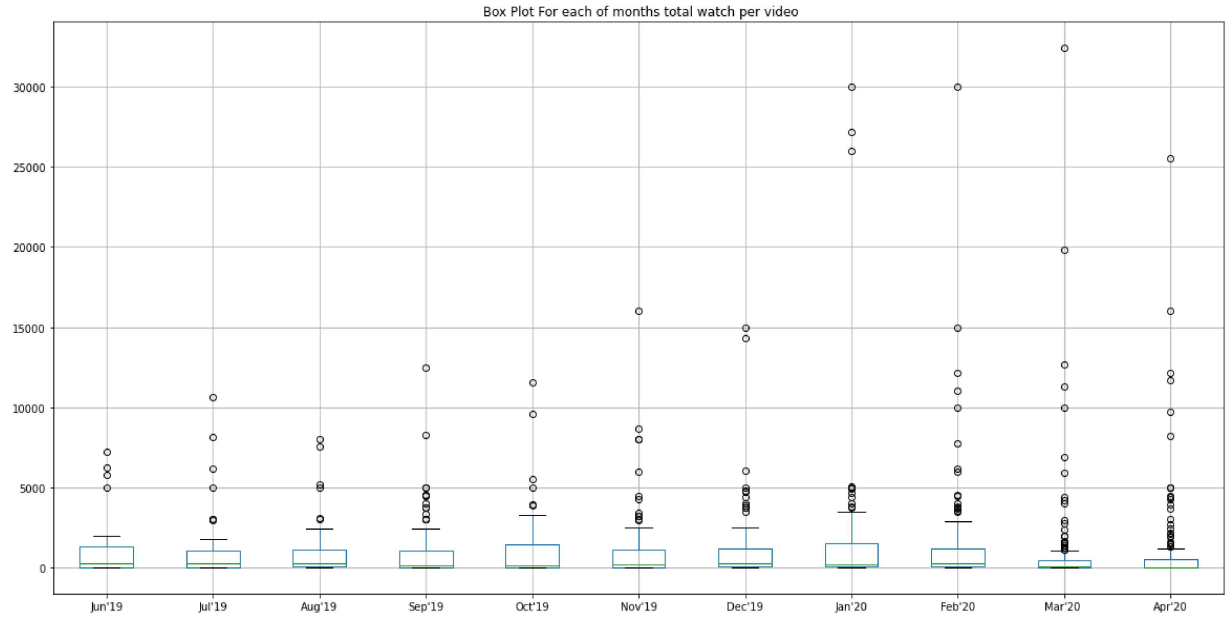


Seems Like watch per videos are decresing with time reason must audience now have more number of options since there is too many courses to buy.

**Also we should focus on bringing more cutomers or it may have some adverse effects on our content creators

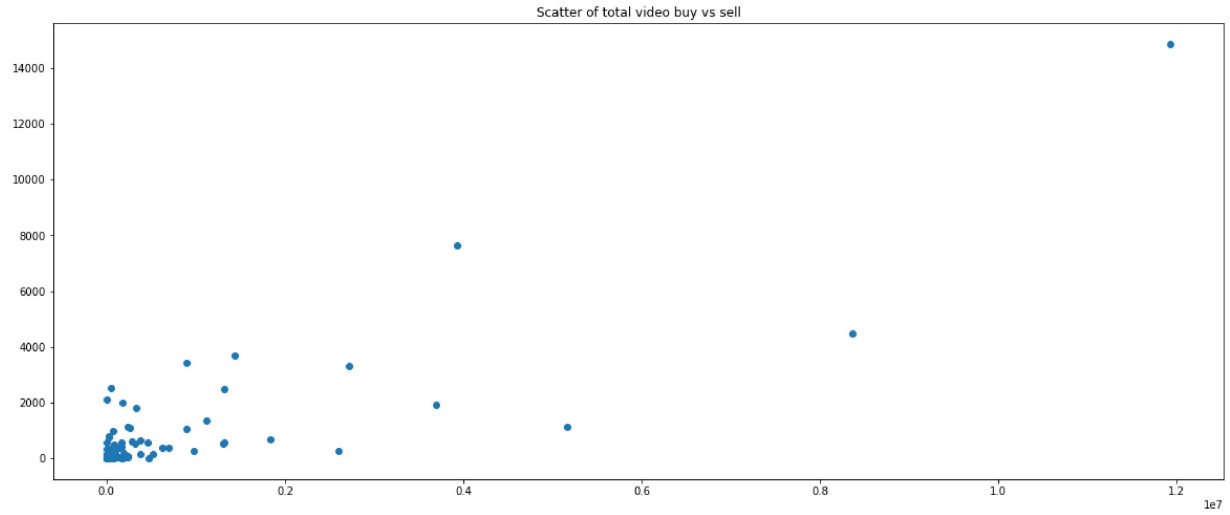
In [44]:
```
df1_sorted = df1.sort_values('Grand Total')
```

In [45]:
```
plt.figure(figsize = (20,10))
df_frqnc[colms].boxplot(grid=True, fontsize=10)
plt.title(" Box Plot For each of months total watch per video")
plt.show()
```



Scenario is also same here

In [46]:
```
plt.figure(figsize = (20,8))
plt.title("Scatter of total video buy vs sell")
plt.scatter(df2['Grand Total'],df1['Grand Total'])

plt.show()
```



The relation seems to be linear

RFM Analysis

In [47]:
```
# Function to get recency for creator
def get_index(l):
    l = list(l)
```

```
    a = 0
    for i in range(len(l)):
        if l[i]!= 0: a = i
    return a
```

In [48]:
```
# To Get Last date of content creation
r = []
for i in range(len(df1)):
    index = get_index(df1.iloc[i,1:-1])
    rd = 11 - index
    r.append(rd)
```

In [49]:
```
def get_rank(x, l):
    l_25, l_50, l_75, l_100 = np.percentile(l,25),np.percentile(l,50), np.percentile
    if x <= l_25 : return 4
    elif x > l_25 and x <= l_50: return 3
    elif x> l_50 and x<= l_75: return 2
    else: return 1
```

In [50]:
```
df_rfm = pd.DataFrame(zip(df1['OrgId'],r,df1['Grand Total'],df2['Grand Total']),colu
```

In [51]:
```
r_score = []

for i in df_rfm['Last Update']:
    if i <= 2:
        r_score.append(1)
        continue
    elif i >2 and i <= 4:
        r_score.append(2)
        continue
    elif i > 4 and i <=7:
        r_score.append(3)
        continue
    else : r_score.append(4)
```

In [52]:
```
f_score = [get_rank(i, df_rfm['Total Videos']) for i in df_rfm['Total Videos'] ]
m_score = [get_rank(i, df_rfm['GMV']) for i in df_rfm['GMV'] ]
```

In [53]:
```
df_rfm['F_Score'] = f_score
df_rfm['M_Score'] = m_score
df_rfm['R_Score'] = r_score

#df_rfm = df_rfm.drop(columns='RFM')
df_rfm = df_rfm.astype({'R_Score':'string','F_Score':'string','M_Score':'string'})
df_rfm['RFM'] = df_rfm.R_Score + df_rfm.F_Score + df_rfm.M_Score

df_rfm
```

Out[53]:

	Id	Last Update	Total Videos	GMV	F_Score	M_Score	R_Score	RFM
0	C1	2	3	600.000012	4	3	1	143
1	C2	1	7	6939.000057	3	2	1	132
2	C3	1	71	367.999990	2	3	1	123
3	C4	1	7	11158.999637	3	2	1	132
4	C5	3	10	8002.000090	3	2	2	232
...	...	...	...	...	...	...	...	...

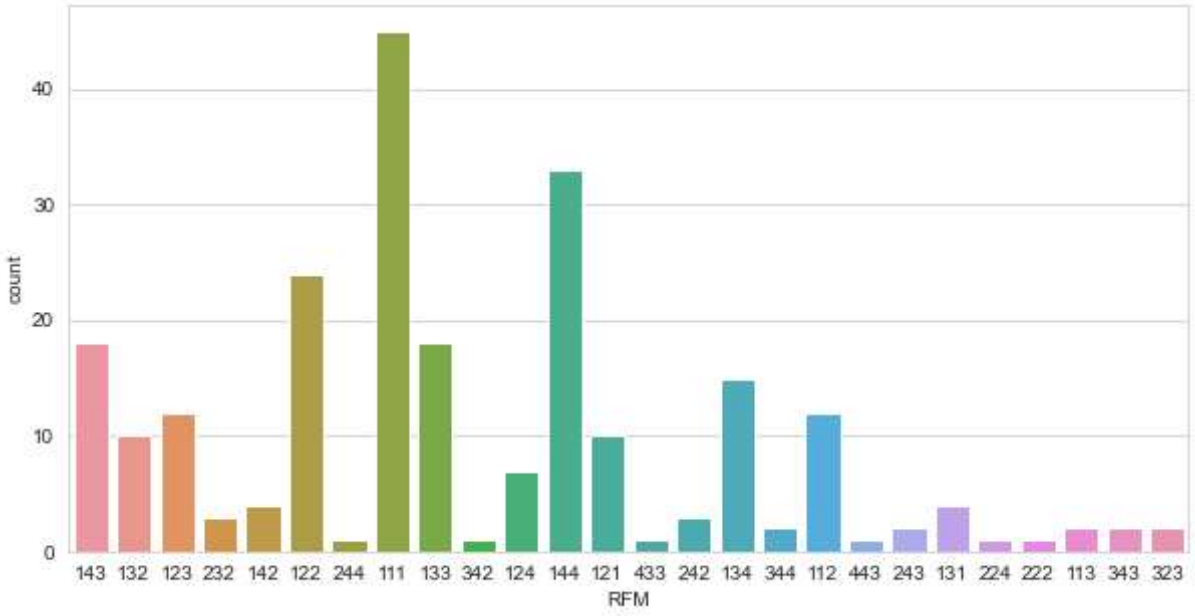
	Id	Last Update	Total Videos	GMV	F_Score	M_Score	R_Score	RFM
229	C230	1	17	34.169999	3	4	1	134
230	C231	2	3	1240.000030	4	3	1	143
231	C232	1	5	157.009998	4	3	1	143
232	C233	1	83	8362.999922	2	2	1	122
233	C234	1	48	8142.000099	2	2	1	122

234 rows × 8 columns

In [54]:
```
len(np.unique(df_rfm.RFM))
```

Out[54]: 26

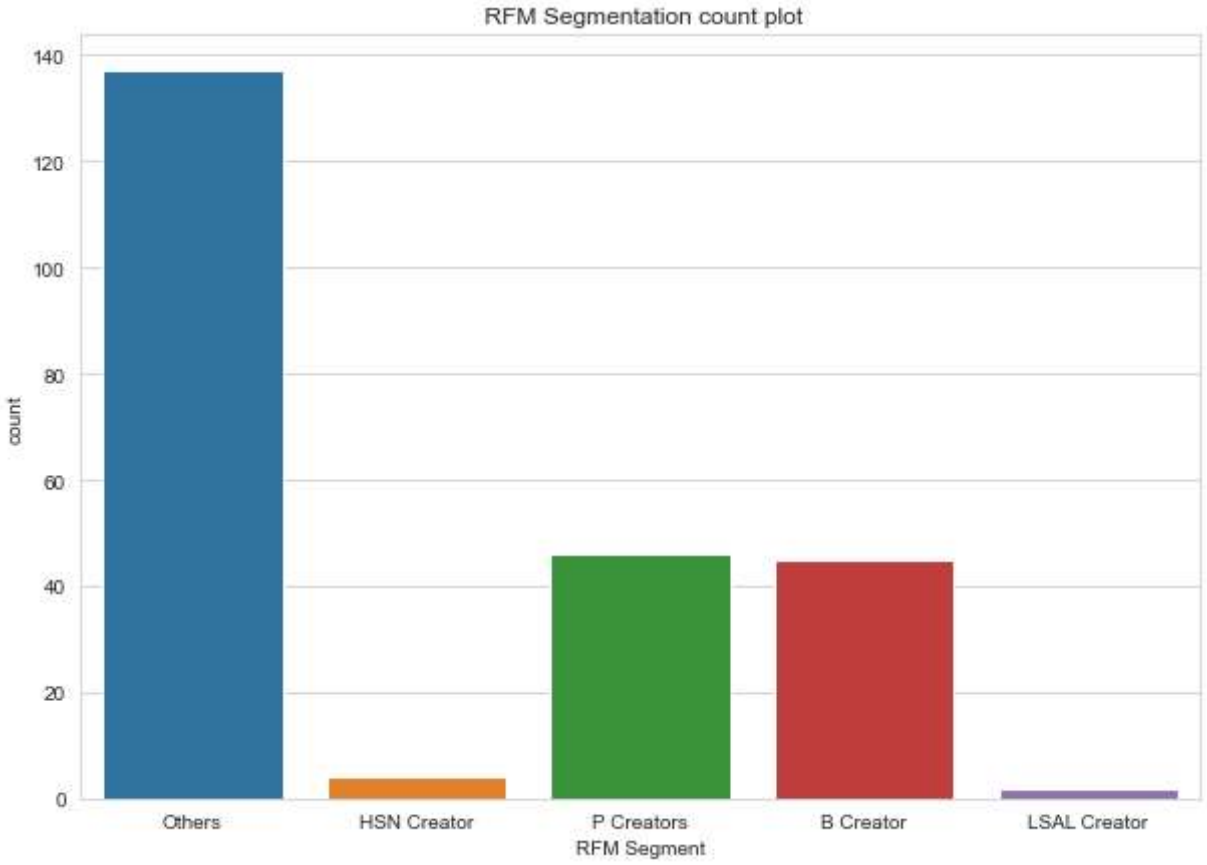
In [374…
```
plt.figure(figsize= (10,5))
sns.countplot(x = 'RFM', data = df_rfm)
plt.show()
```



In [57]:
```
def rfm_seg(x):

    x = int(x)
    if x == 111 : return 'B Creator' # Best Creators
    elif x in [112,121,122] : return 'P Creators' # Potentil creators
    elif x == 141 or x == 142 : return 'HSN Creator' #Highly sold new creators
    elif x == 113 or x == 114 : return 'LSAL Creator'#Low sell active loyal creator
    elif x in [411, 412, 421, 422] : return 'Cb Creator' #Churned Best Creators
    else : return 'Others'
```

In [58]:
```
df_rfm['RFM Segment'] =  df_rfm.RFM.apply(rfm_seg)
```

In [373…
```
plt.figure(figsize= (10,7))
sns.countplot(x = 'RFM Segment', data = df_rfm)
plt.title(" RFM Segmentation count plot")
plt.show()
```



In [355…
```
df_rfm['RFM Segment'].value_counts()
```

Out[355…
```
Others          137
P Creators       46
B Creator        45
HSN Creator       4
LSAL Creator      2
Name: RFM Segment, dtype: int64
```

(B Creator)Best Creators :– Communications with this group should make them feel valued and appreciated. These creators likely generate a disproportionately high percentage of overall revenues and thus focusing on keeping them happy should be a top priority. Further analyzing their individual preferences and affinities will provide additional opportunities for even more personalized messaging. (HSN Creator) High - Earning New Creators :– It is always a good idea to carefully "incubate" all new creators, but because these new creators spent a lot on their first purchase, it's even more important. Like with the Best Customers group, it's important to make them feel valued and appreciated – and to give them terrific incentives to continue interacting with the brand. (LSAL Creator) Lowest-Profitable Active Loyal Creators (At Risk creator) :– These frequent creators are active and loyal, but they are low soldout. Marketers should create campaigns for this group that make them feel valued, and incentivize them to increase their spend levels. As loyal customers, it often also pays to reward them with special offers if they spread the word about the brand to their friends, e.g., via social networks. (P Creators) Potential creators :- They are quite promising and high in number. Revenues generated by them is in range 50 - 75 percentile which is not bad, they have good potential of going up in the lader, communcating and undestanding them could be helpful for both of us. (Cb Creator) Churned Best creators – These are valuable creators who stopped transacting a long time ago. While it's often challenging to re-engage churned creators, the high value of these creators makes it worthwhile trying. Like with the Best creators group, it's important to communicate with them on the basis of their specific preferences, as known from earlier transaction data.

In [75]:
```
del df
df = df1.merge(df2, on= 'OrgId' )
df = df.merge(df_frqnc, on = 'OrgId')
df = df.drop(columns=['Grand Total','Grand Total_y','Grand Total_x'])
df = df.fillna(0)
df
```

Out[75]:

OrgId	Jun'19_x	Jul'19_x	Aug'19_x	Sep'19_x	Oct'19_x	Nov'19_x	Dec'19_x	Jan'20_x	Feb'2(

	OrgId	Jun'19_x	Jul'19_x	Aug'19_x	Sep'19_x	Oct'19_x	Nov'19_x	Dec'19_x	Jan'20_x	Feb'2
0	C1	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	C2	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	
2	C3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	C4	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	
4	C5	0.0	1.0	1.0	0.0	0.0	7.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
229	C230	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
230	C231	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	
231	C232	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	
232	C233	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
233	C234	0.0	0.0	0.0	1.0	0.0	0.0	4.0	1.0	

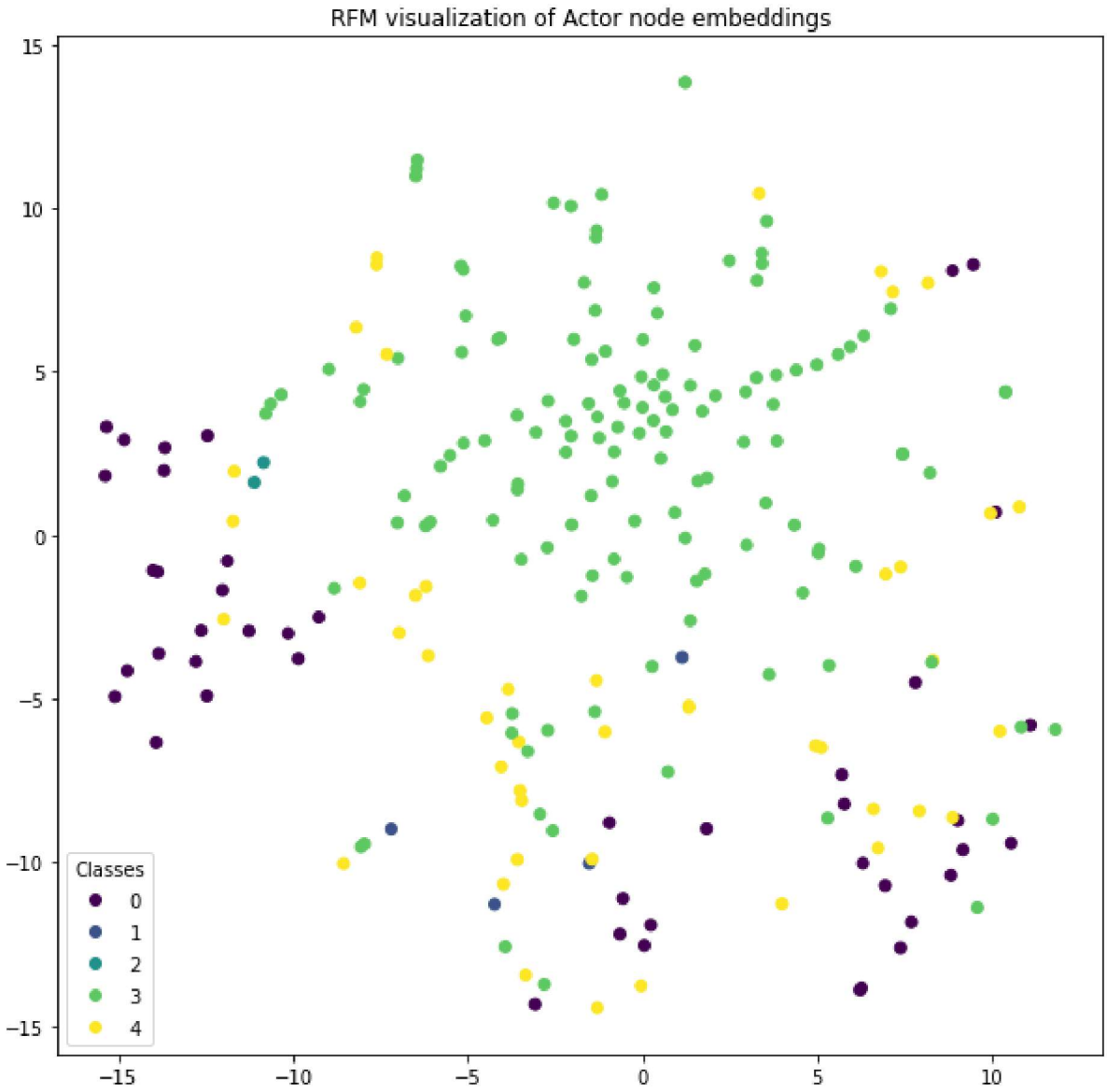
234 rows × 34 columns

In [76]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
data = scaler.fit_transform(df.drop(columns='OrgId'))
```

In [77]:
```
from sklearn.manifold import TSNE
transform = TSNE

trans = transform(n_components=2)
creator_embeddings_2d = trans.fit_transform(data)
```

In [ ]:
```
sns.scatterplot(data=tips, x="total_bill", y="tip", hue="time")
```

In [99]:
```
import numpy as np
# draw the points

label_map = { l: i for i, l in enumerate(np.unique(df_rfm['RFM Segment']))}
node_colours = [ label_map[target] for target in df_rfm['RFM Segment']]


fig, ax = plt.subplots(figsize = (10,10))

#plt.axes().set(aspect="equal")
scatter = ax.scatter(creator_embeddings_2d[:,0],
            creator_embeddings_2d[:,1],
            alpha=1,c = node_colours)




# produce a legend with the unique colors from the scatter
legend1 = ax.legend(*scatter.legend_elements(),
                    loc="lower left", title="Classes")

plt.title('RFM visualization of Actor node embeddings')

plt.show()
```



In [84]:
```
label_map
```

Out[84]:
```
{'B Creator': 0,
 'HSN Creator': 1,
 'LSAL Creator': 2,
 'Others': 3,
 'P Creators': 4}
```

Segmentation using unsupervised machine learning like K-Means and DB Scan

K-Means Clustering

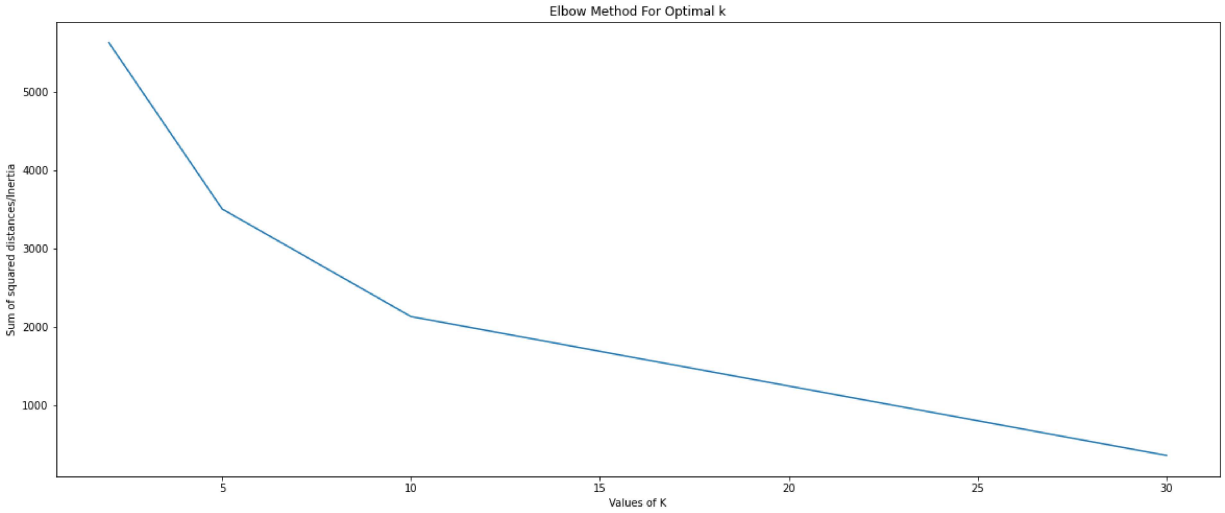
In [92]:
```
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score
import warnings
warnings.filterwarnings("ignore")
```

In [93]:
```
from tqdm import tqdm
inertia = []
k = [2, 5, 10, 30]
```

```
for number_of_cluster in tqdm(k):

    kmeans = KMeans(n_clusters=number_of_cluster, init='k-means++', max_iter=300, n_

    kmeans.fit(data)

    inertia.append(kmeans.inertia_)
```

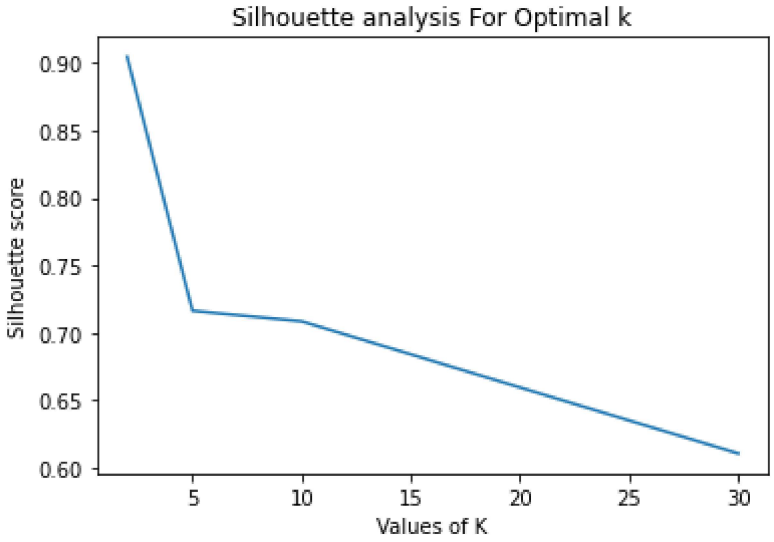
```
100%|██████████| 4/4 [00:00<00:00, 22.25it/s]
```

In [94]:
```
plt.figure(figsize = (20,8))
sns.lineplot(k , inertia)
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
plt.show()
```



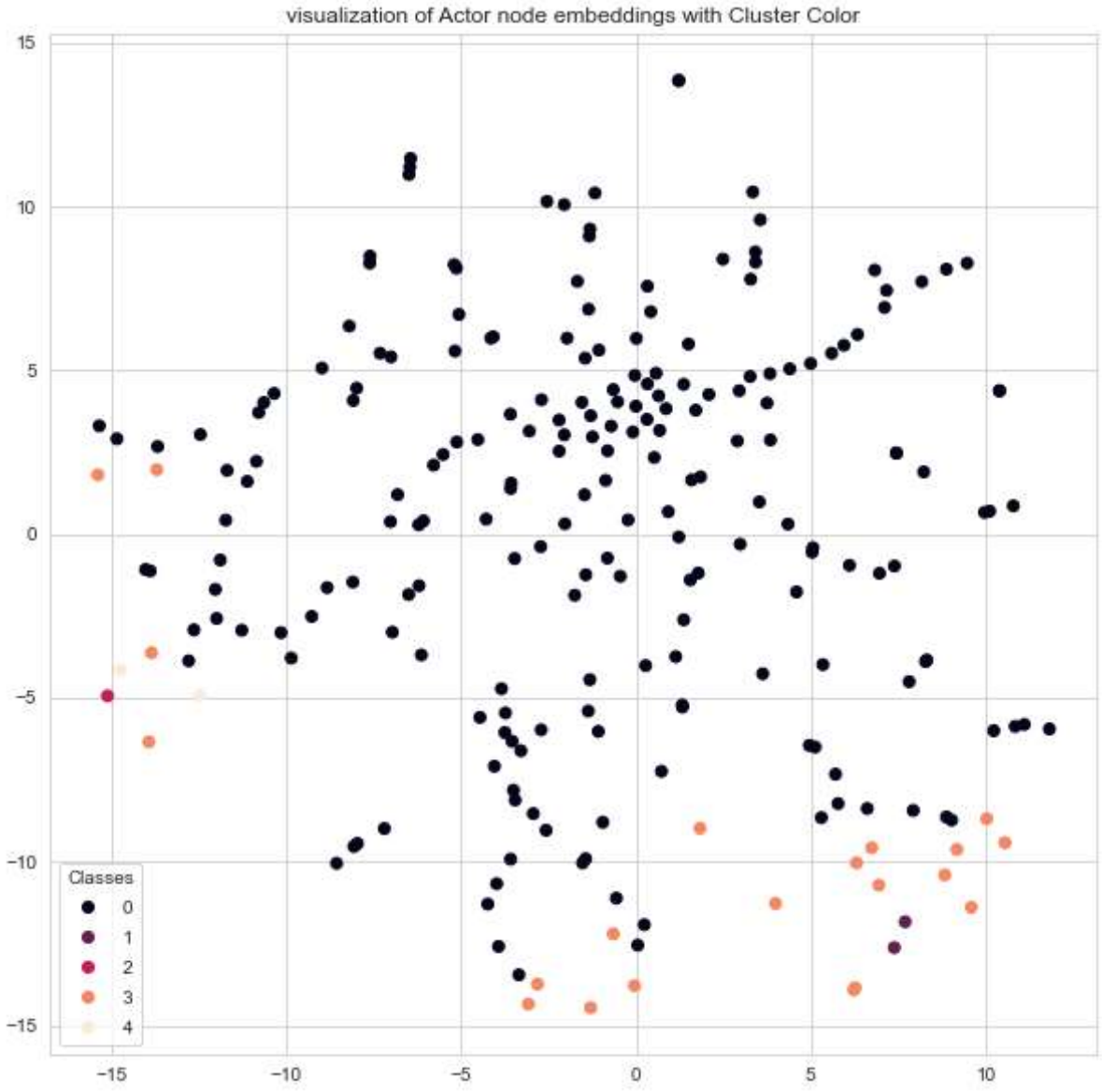
In [95]:
```
silhouette_avg = []
k = [2, 5, 10, 30]

for number_of_cluster in tqdm(k):

    kmeans = KMeans(n_clusters=number_of_cluster, init='k-means++', max_iter=300, n_

    kmeans.fit(data)

    inertia.append(kmeans.inertia_)

    silhouette_avg.append(silhouette_score(data, kmeans.labels_))

sns.lineplot(k , silhouette_avg)

plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()
```

```
100%|██████████| 4/4 [00:00<00:00, 18.34it/s]
```



In [96]:
```
kmeans_act = KMeans(n_clusters= 5, init='k-means++', max_iter=300, n_init=10, random

kmeans_act.fit(data)

index_cluster = kmeans_act.labels_
```

In [351…
```
label_map = { l: i for i, l in enumerate(np.unique(kmeans_act.labels_))}
node_colours = [ label_map[target] for target in kmeans_act.labels_]


fig, ax = plt.subplots(figsize = (10,10))

scatter = ax.scatter(creator_embeddings_2d[:,0],
            creator_embeddings_2d[:,1],
            alpha=1,c = node_colours)

legend1 = ax.legend(*scatter.legend_elements(),
                    loc="lower left", title="Classes")


plt.title('visualization of Actor node embeddings with Cluster Color')

plt.show()
```



In [352…
```
label_map
```

Out[352…
```
{0: 0, 1: 1, 2: 2, 3: 3, 4: 4}
```

In [108…
```
df_rfm['K_means Clusters'] = kmeans_act.labels_
```

In [111…
```
df_rfm[df_rfm['K_means Clusters'] == 0]['RFM Segment'].value_counts()
```

Out[111…
```
Others          134
P Creators       42
B Creator        26
HSN Creator       4
LSAL Creator      2
Name: RFM Segment, dtype: int64
```

In [117…
```
df_rfm[df_rfm['K_means Clusters'] == 1]
```

Out[117…

	Id	Last Update	Total Videos	GMV	F_Score	M_Score	R_Score	RFM	RFM Segment	K_means Clusters
25	C26	1	1120	5.164276e+06	1	1	1	111	B Creator	1
144	C145	1	3314	2.713225e+06	1	1	1	111	B Creator	1

In [118…
```
df_rfm[df_rfm['K_means Clusters'] == 2]
```

Out[118...

	Id	Last Update	Total Videos	GMV	F_Score	M_Score	R_Score	RFM	RFM Segment	K_means Clusters
184	C185	1	14865	1.193064e+07	1	1	1	111	B Creator	2

In [124...
```
df_rfm[df_rfm['K_means Clusters'] == 3]['RFM Segment'].value_counts()
```

Out[124...
```
B Creator     14
P Creators     4
Others         3
Name: RFM Segment, dtype: int64
```

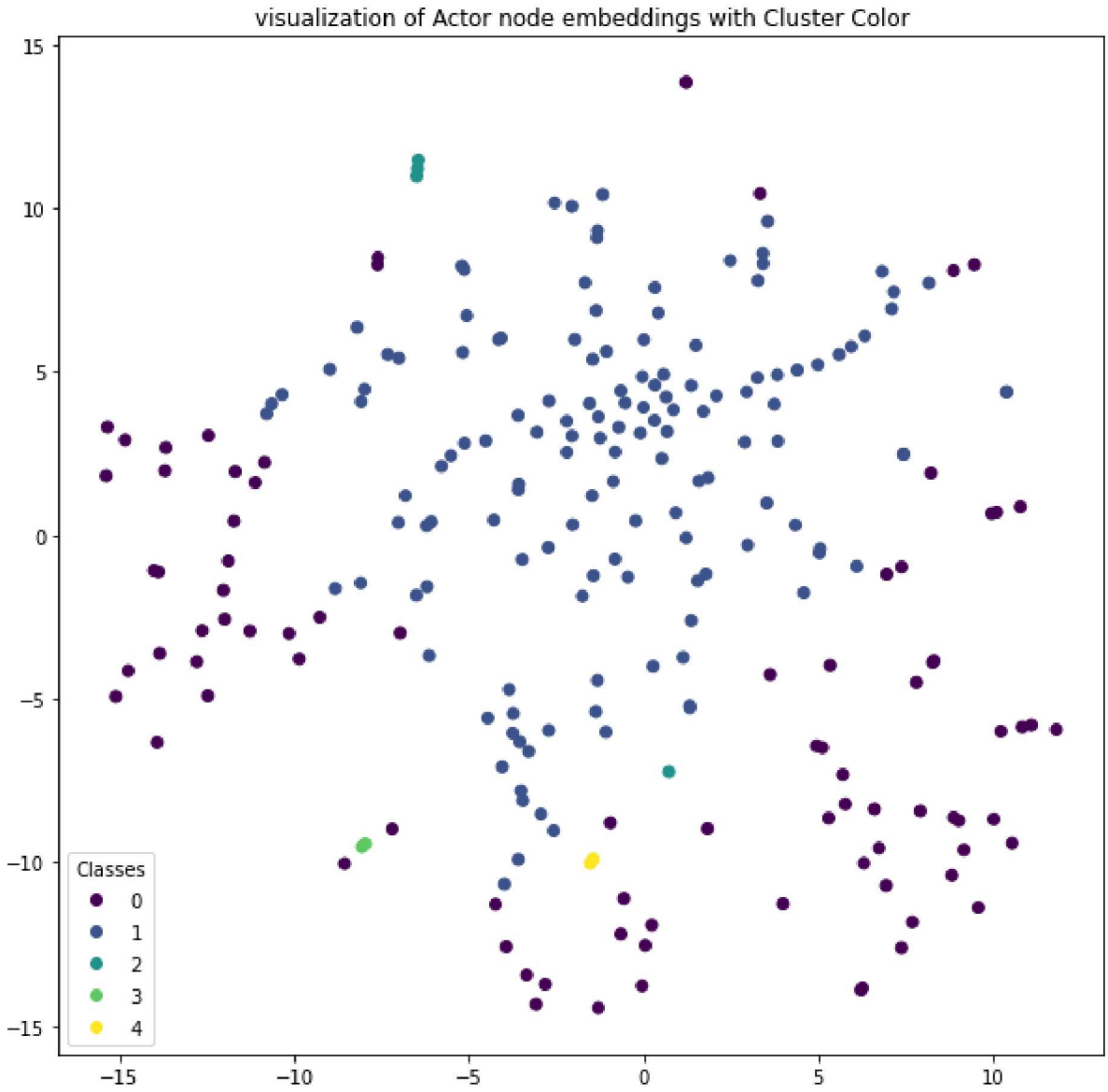
In [120...
```
df_rfm[df_rfm['K_means Clusters'] == 4]
```

Out[120...

	Id	Last Update	Total Videos	GMV	F_Score	M_Score	R_Score	RFM	RFM Segment	K_means Clusters
22	C23	1	7640	3.924930e+06	1	1	1	111	B Creator	4
133	C134	1	4466	8.358156e+06	1	1	1	111	B Creator	4

DBSCAN Clustering

In [312...
```
dbscan = DBSCAN(eps = .445,min_samples=2) # Manually I have checked with othe hyper

dbscan.fit(data)

index_cluster = dbscan.labels_

label_map = { l: i for i, l in enumerate(np.unique(index_cluster))}
node_colours = [ label_map[target] for target in index_cluster]


fig, ax = plt.subplots(figsize = (10,10))

scatter = ax.scatter(creator_embeddings_2d[:,0],
            creator_embeddings_2d[:,1],
            alpha=1,c = node_colours)

legend1 = ax.legend(*scatter.legend_elements(),
                    loc="lower left", title="Classes")


plt.title('visualization of Actor node embeddings with Cluster Color')

plt.show()
```



label_map : {-1: 0, 0: 1, 1: 2, 2: 3, 3: 4}

In [313…
```
label_map
```

Out[313…
```
{-1: 0, 0: 1, 1: 2, 2: 3, 3: 4}
```

In [314…
```
df_rfm['DB Clusters'] = index_cluster
df_rfm['DB Clusters'].value_counts()
```

Out[314…
```
 0    140
-1     86
 1      4
 2      2
 3      2
Name: DB Clusters, dtype: int64
```

In [305…
```
df_rfm[df_rfm['DB Clusters'] == 0]['RFM Segment'].value_counts()
```

Out[305…
```
Others          118
P Creators       21
HSN Creator       1
Name: RFM Segment, dtype: int64
```

In [357…
```
df_rfm[df_rfm['DB Clusters'] == -1]['RFM Segment'].value_counts()
```

Out[357…
```
B Creator       45
P Creators      24
```

```
Others          13
HSN Creator      2
LSAL Creator     2
Name: RFM Segment, dtype: int64
```

In [310…
```
#total revenue percent from cluster -1
df_rfm[df_rfm['DB Clusters'] == -1].GMV.sum()/df_rfm.GMV.sum()
```

Out[310… 0.9918834879064642

In [308…
```
#total video creation percent from cluster -1
df_rfm[df_rfm['DB Clusters'] == -1]['Total Videos'].sum()/df_rfm['Total Videos'].sum
```

Out[308… 0.9616318008649792

In [297…
```
#total percent of population from cluster -1
len(df_rfm[df_rfm['DB Clusters'] == -1])/df_rfm['Total Videos'].shape[0]
```

Out[297… 0.38461538461538464

**Seems like these 38 percent of people who are the real asset of the company

In [322…
```
db_1 = df_rfm[df_rfm['DB Clusters'] == -1]
for i in range(0,101,10):
    v = np.percentile(db_1['GMV'],i)
    print("{}'th Percentile value of grand total: {}".format(i,v))
```

```
0'th Percentile value of grand total: 367.0099913086739
10'th Percentile value of grand total: 6074.005092444818
20'th Percentile value of grand total: 12006.029908306882
30'th Percentile value of grand total: 27821.000323895318
40'th Percentile value of grand total: 56035.7888097389
50'th Percentile value of grand total: 89836.52066674069
60'th Percentile value of grand total: 173456.04815726707
70'th Percentile value of grand total: 274671.5029542076
80'th Percentile value of grand total: 627508.0062083553
90'th Percentile value of grand total: 1376932.2431501406
100'th Percentile value of grand total: 11930639.626307186
```

In [323…
```
db_1 = df_rfm[df_rfm['DB Clusters'] == -1]
for i in range(0,101,10):
    v = np.percentile(db_1['Total Videos'],i)
    print("{}'th Percentile value of grand total: {}".format(i,v))
```

```
0'th Percentile value of grand total: 3.0
10'th Percentile value of grand total: 14.0
20'th Percentile value of grand total: 22.0
30'th Percentile value of grand total: 51.0
40'th Percentile value of grand total: 116.0
50'th Percentile value of grand total: 194.5
60'th Percentile value of grand total: 371.0
70'th Percentile value of grand total: 559.5
80'th Percentile value of grand total: 986.0
90'th Percentile value of grand total: 2052.5
100'th Percentile value of grand total: 14865.0
```

Some Stats about label 0

In [319…
```
#total revenue percent from cluster 0
df_rfm[df_rfm['DB Clusters'] == 0].GMV.sum()/df_rfm.GMV.sum()
```

Out[319… 0.006132297026545686

In [375…
```
df_rfm[df_rfm['DB Clusters'] == 0]['Total Videos'].sum()/df_rfm['Total Videos'].sum(
```

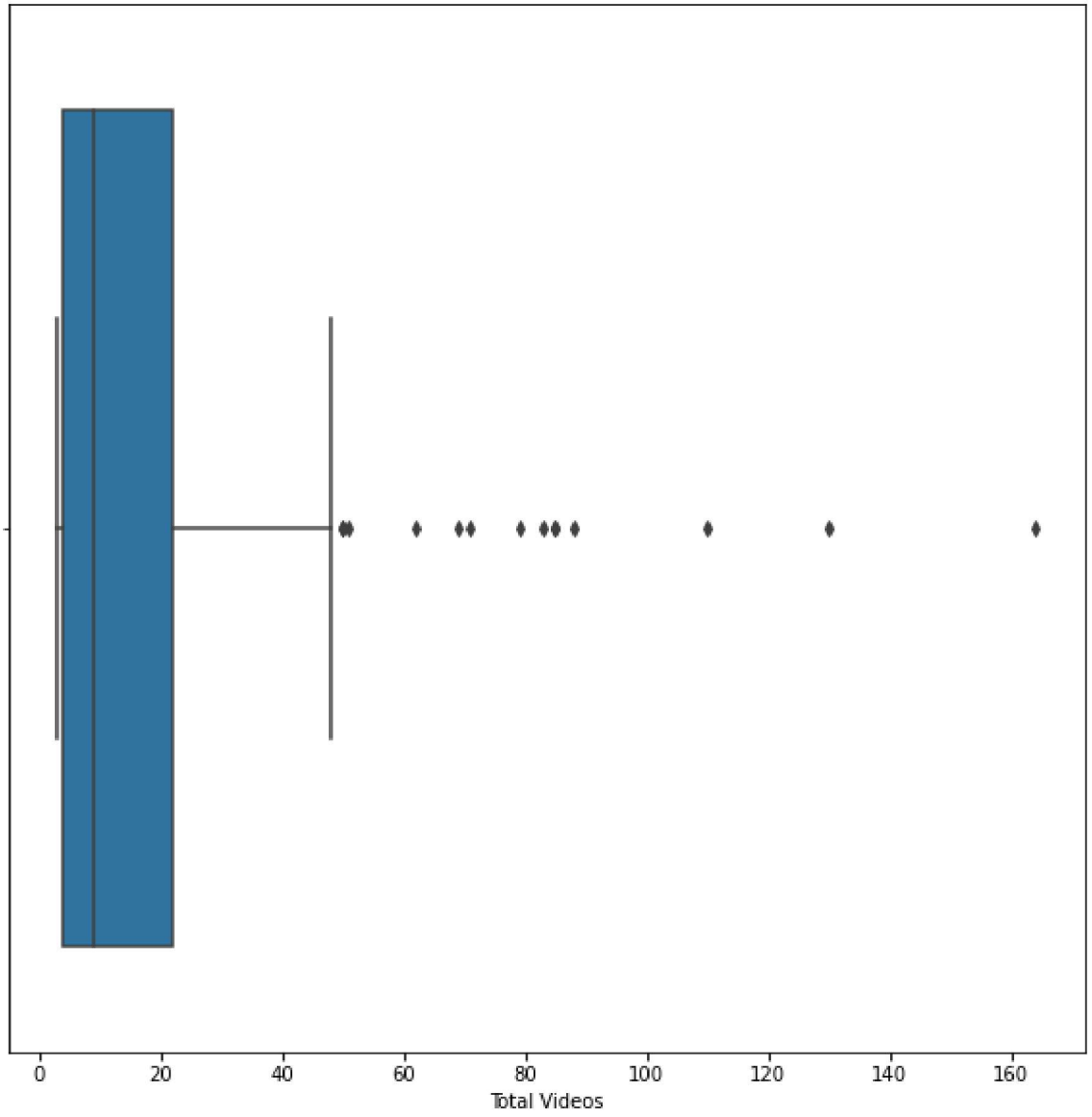
Out[375…
```
0.03750050841253271
```

In [320…
```
#total percent of population from cluster -1
len(df_rfm[df_rfm['DB Clusters'] == 0])/df_rfm['Total Videos'].shape[0]
```

Out[320…
```
0.5982905982905983
```

In [324…
```
plt.figure(figsize=(10,10))

sns.boxplot(df_rfm[df_rfm['DB Clusters'] == 0]['Total Videos'])
```

Out[324…
```
<AxesSubplot:xlabel='Total Videos'>
```



In [326…
```
db_2 = df_rfm[df_rfm['DB Clusters'] == 0]
for i in range(0,101,10):
    v = np.percentile(db_2['GMV'],i)
    print("{}'th Percentile value of grand total: {}".format(i,v))
```

```
0'th Percentile value of grand total: 2.999999916180962
10'th Percentile value of grand total: 6.02999985963106
20'th Percentile value of grand total: 14.016000141203367
30'th Percentile value of grand total: 38.25100086703879
40'th Percentile value of grand total: 79.79999977722758
50'th Percentile value of grand total: 252.010006330907
```

```
60'th Percentile value of grand total: 460.20400098450375
70'th Percentile value of grand total: 958.1120046988117
80'th Percentile value of grand total: 2075.199972915639
90'th Percentile value of grand total: 6560.708861170153
100'th Percentile value of grand total: 37512.06085802613
```

In [327…
```
df_rfm[df_rfm['DB Clusters'] == 0]['RFM Segment'].value_counts()
```

Out[327…
```
Others         118
P Creators      21
HSN Creator      1
Name: RFM Segment, dtype: int64
```

And these 60% of people belonging to cluster 0 are least important creators. They don't even contribute 1% of profit

**Surprisingly there is one new high earning creator is among them so we can talk to him, and some potential creators to improve their level

FINAL SEGMENTED DATA FRAME

In [330…
```
df_rfm
```

Out[330…

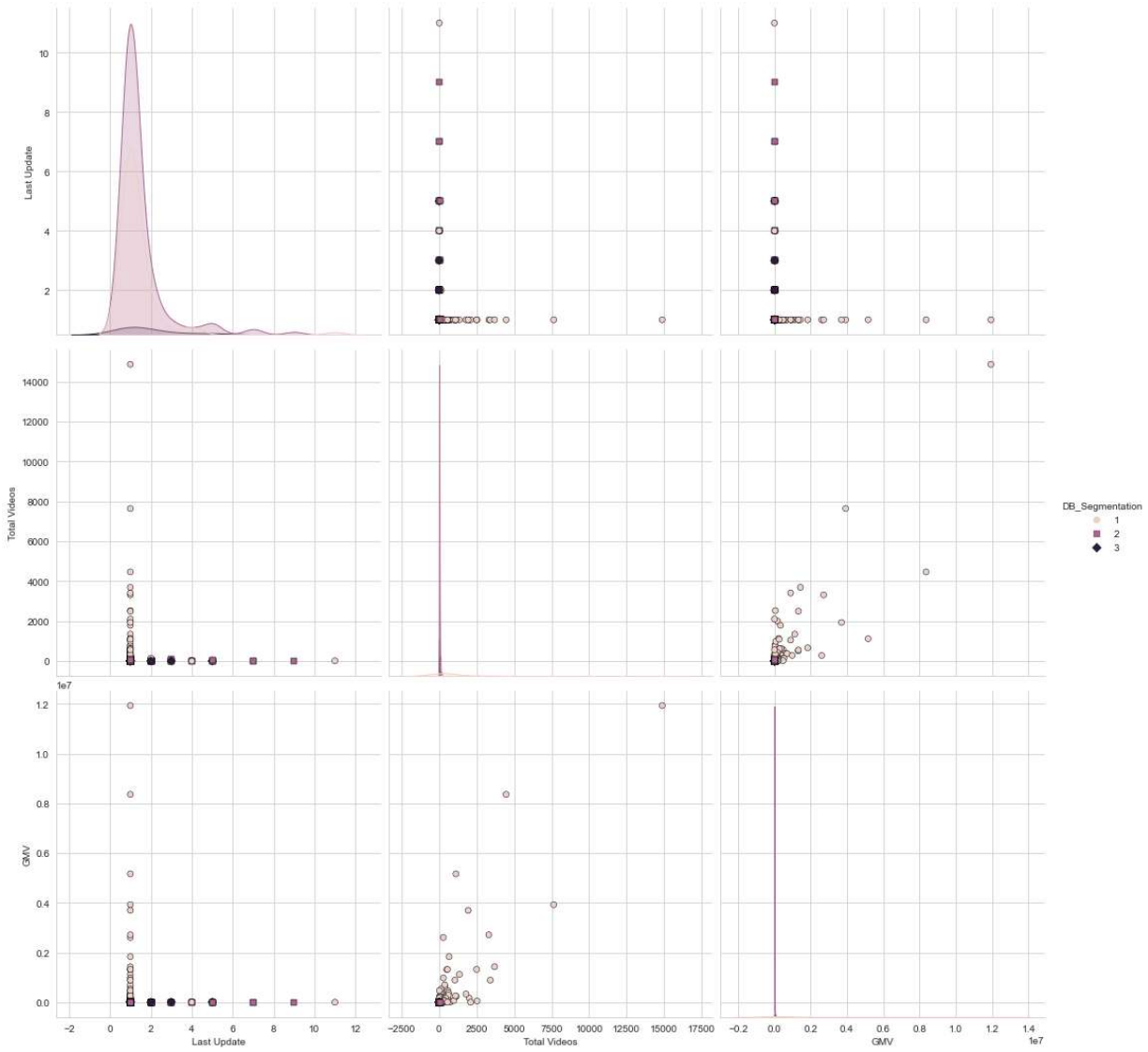
	Id	Last Update	Total Videos	GMV	F_Score	M_Score	R_Score	RFM	RFM Segment	K_means Clusters	Cl
0	C1	2	3	600.000012	4	3	1	143	Others	0	
1	C2	1	7	6939.000057	3	2	1	132	Others	0	
2	C3	1	71	367.999990	2	3	1	123	Others	0	
3	C4	1	7	11158.999637	3	2	1	132	Others	0	
4	C5	3	10	8002.000090	3	2	2	232	Others	0	
…	…	…	…	…	…	…	…	…	…	…	…
229	C230	1	17	34.169999	3	4	1	134	Others	0	
230	C231	2	3	1240.000030	4	3	1	143	Others	0	
231	C232	1	5	157.009998	4	3	1	143	Others	0	
232	C233	1	83	8362.999922	2	2	1	122	P Creators	0	
233	C234	1	48	8142.000099	2	2	1	122	P Creators	0	

234 rows × 11 columns

In [339…
```
def redist(x):
    if x == -1 : return 1
    elif x == 0 : return 2
    else: return 3
```

In [341…
```
df_rfm['DB_Segmentation'] = df_rfm['DB Clusters'].apply(redist)
```

In [349…
```
sns.pairplot(df_rfm[['Last Update','Total Videos','GMV','DB_Segmentation']],
             hue = 'DB_Segmentation',markers=["o", "s", "D"],plot_kws = {'edgecolor'
plt.show()
```



In [376…
```
df_rfm.to_csv("ClassPlus_Segmented Dataset.csv")
```

In [ ]:
```

```