In [61]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
```

In [20]:
```
voice_usage_info = pd.read_csv("data/VOICE_INCOMING_CALL_USAGE_OFFNET.csv")

voice_usage_info.shape
```

Out[20]:
```
(36156, 10)
```

In [21]:
```
voice_usage_info
```

Out[21]:

	event_source	event_source.hash	call_count	duration (seconds)	ym	rtom_code	location_code	msan	equp_i
0	412244863	ef74b1171331411a4d2e9fada9158215	2	186	202001	R-MH	MH-POL-NODE	MSAG5200-ISL	4914
1	412244863	ef74b1171331411a4d2e9fada9158215	3	114	202002	R-MH	MH-POL-NODE	MSAG5200-ISL	4914
2	662227788	30576ca16275a38f6bb4491015edb96c	10	2435	201909	R-MT	MT-AVR-NODE	MSAG5200-ISL	4671
3	412244863	ef74b1171331411a4d2e9fada9158215	14	1061	202003	R-MH	MH-POL-NODE	MSAG5200-ISL	4914
4	412228632	d3439750c4842a755ebfddc64a0b42ae	65	7971	201910	R-MH	MH-POL-NODE	MSAG5200-ISL	4914
...	...	...	...	...	...	...	...	...	
36151	112411111	64879716f309303245754526212d990e	182	20134	202008	R-KON	IDH-KGW-NODE	MSAG5200-ISL	61944
36152	112411111	64879716f309303245754526212d990e	173	14409	201908	R-KON	IDH-KGW-NODE	MSAG5200-ISL	61944
36153	112411111	64879716f309303245754526212d990e	196	19571	201910	R-KON	IDH-KGW-NODE	MSAG5200-ISL	61944
36154	112411111	64879716f309303245754526212d990e	205	17685	201911	R-KON	IDH-KGW-NODE	MSAG5200-ISL	61944
36155	112411111	64879716f309303245754526212d990e	242	19340	201912	R-KON	IDH-KGW-NODE	MSAG5200-ISL	61944

36156 rows × 10 columns

In [22]:
```
# ToDo
# Handle null values
# Drop unnecessary cols
# Pivot by monthYear
# Usage rating Upload/Downlad/Duration - bucketizing/ add rating cols
# location grouping/bucketizing (consider 4 location cols)
# User profile dim table
# visualize and analyze data
```

In [23]:
```
voice_usage_info.drop_duplicates("event_source")

voice_usage_info.shape
```

Out[23]: (36156, 10)

In [24]:
```
voice_usage_info = voice_usage_info.drop(['event_source.hash'], axis=1)
voice_usage_info.rename(columns={'duration (seconds)': 'duration','ym':'year-month'}, inplace=True)
voice_usage_info
```

Out[24]:

	event_source	call_count	duration	year-month	rtom_code	location_code	msan	equp_id	equp_index
0	412244863	2	186	202001	R-MH	MH-POL-NODE	MSAG5200-ISL	49145	1
1	412244863	3	114	202002	R-MH	MH-POL-NODE	MSAG5200-ISL	49145	1
2	662227788	10	2435	201909	R-MT	MT-AVR-NODE	MSAG5200-ISL	46716	1
3	412244863	14	1061	202003	R-MH	MH-POL-NODE	MSAG5200-ISL	49145	1
4	412228632	65	7971	201910	R-MH	MH-POL-NODE	MSAG5200-ISL	49145	1
...	...	...	...	...	...	...	...	...	...
36151	112411111	182	20134	202008	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1
36152	112411111	173	14409	201908	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1
36153	112411111	196	19571	201910	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1
36154	112411111	205	17685	201911	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1
36155	112411111	242	19340	201912	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1

36156 rows × 9 columns

In [25]:
```
voice_usage_info['duration'].fillna(0)
voice_usage_info['call_count'].fillna(0)
```

Out[25]:
```
0          2
1          3
2         10
3         14
4         65
        ...
36151    182
36152    173
36153    196
36154    205
36155    242
Name: call_count, Length: 36156, dtype: int64
```

In [26]:
```
voice_usage_info.isnull().sum(axis=0)
```

Out[26]:
```
event_source     0
call_count       0
duration         0
year-month       0
rtom_code        0
location_code    0
msan             0
equp_id          0
equp_index       0
dtype: int64
```

Phase 1

Usage Ranking

- 1. Pivot by year-month
- 2. Bin by Upload/Download/Duration
- 3. Usage Rank
- 4. Labling usage

In [125]:
```
voice_usage = voice_usage_info[['event_source','call_count','duration','year-month']]
voice_usage
```

Out[125]:

	event_source	call_count	duration	year-month
0	412244863	2	186	202001
1	412244863	3	114	202002
2	662227788	10	2435	201909
3	412244863	14	1061	202003
4	412228632	65	7971	201910
...	...	...	...	...
36151	112411111	182	20134	202008
36152	112411111	173	14409	201908
36153	112411111	196	19571	201910
36154	112411111	205	17685	201911
36155	112411111	242	19340	201912

36156 rows × 4 columns

In [126]:
```
pivot_voice_usage_info = pd.pivot_table(voice_usage,index=['event_source'],columns=['year-month'], a
pivot_voice_usage_info
```

Out[126]:

	call_count										...	duration			
year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005	...	201911	201912	202001	202002
event_source															
112053582	72	45	66	82	27	15	0	0	0	0	...	4171	1403	1508	(
112053623	5	0	2	0	0	0	2	1	1	0	...	0	0	0	8(
112053643	281	253	321	264	192	221	237	162	163	196	...	23680	11903	13158	22212
112054356	20	17	16	0	0	0	0	0	0	0	...	0	0	0	(
112055452	17	37	58	25	26	23	12	35	71	23	...	1040	1725	1477	53(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
912286932	99	104	84	93	144	71	54	50	39	25	...	4013	8937	4710	530
912286967	17	38	37	31	42	19	13	4	3	23	...	2574	1646	1680	55(
912286996	4	3	0	0	1	0	3	0	2	1	...	0	57	0	30(
912287030	4	0	0	0	0	4	4	6	6	15	...	0	0	261	12(
912287049	4	2	1	1	3	1	1	1	1	0	...	23	143	37	6(

3329 rows × 26 columns

In [127]:
```
pivot_voice_usage_info.columns
```

Out[127]:
```
MultiIndex([('call_count', 201908),
            ('call_count', 201909),
            ('call_count', 201910),
            ('call_count', 201911),
            ('call_count', 201912),
            ('call_count', 202001),
            ('call_count', 202002),
            ('call_count', 202003),
            ('call_count', 202004),
            ('call_count', 202005),
            ('call_count', 202006),
            ('call_count', 202007),
            ('call_count', 202008),
            (  'duration', 201908),
            (  'duration', 201909),
            (  'duration', 201910),
            (  'duration', 201911),
            (  'duration', 201912),
            (  'duration', 202001),
            (  'duration', 202002),
            (  'duration', 202003),
            (  'duration', 202004),
            (  'duration', 202005),
            (  'duration', 202006),
            (  'duration', 202007),
            (  'duration', 202008)],
           names=[None, 'year-month'])
```

In [128]:
```
pivot_voice_usage_info.describe().apply(lambda s: s.apply('{0:.5f}'.format))
```

Out[128]:

	call_count										...	c
year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005	...	
count	3329.00000	3329.00000	3329.00000	3329.00000	3329.00000	3329.00000	3329.00000	3329.00000	3329.00000	3329.00000	...	
mean	48.44067	47.52598	51.09042	48.65906	51.97236	43.26224	46.80805	43.56083	33.52989	37.42746	...	
std	130.59213	136.98717	145.82699	131.81204	133.12486	125.44991	133.14003	103.47912	86.67630	90.99460	...	
min	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	...	
25%	4.00000	4.00000	5.00000	4.00000	5.00000	4.00000	4.00000	4.00000	1.00000	3.00000	...	
50%	23.00000	24.00000	25.00000	24.00000	26.00000	22.00000	23.00000	21.00000	14.00000	17.00000	...	
75%	57.00000	56.00000	60.00000	58.00000	61.00000	51.00000	54.00000	52.00000	38.00000	44.00000	...	
max	4459.00000	4940.00000	5242.00000	4397.00000	4162.00000	4728.00000	4506.00000	2723.00000	1873.00000	2708.00000	...	4

8 rows × 26 columns

In [129]:
```
voice_usage.describe().apply(lambda s: s.apply('{0:.5f}'.format))
```

Out[129]:

	event_source	call_count	duration	year-month
count	36156.00000	36156.00000	36156.00000	36156.00000
mean	334648410.45956	54.83004	5550.79423	201968.02716
std	206951545.50293	134.93060	12798.75523	46.04741
min	112053582.00000	1.00000	1.00000	201908.00000
25%	112809457.00000	11.00000	823.00000	201911.00000
50%	332284754.00000	30.00000	2667.00000	202002.00000
75%	512232254.00000	62.00000	6400.00000	202005.00000
max	912287049.00000	5558.00000	709004.00000	202008.00000

In [130]:
```
# binning call_count
count_bins = [0, 11, 30, 100,300,800,2000,3500, 5558]
for col in pivot_voice_usage_info['call_count'].columns:
    pivot_voice_usage_info['call_count'][col] = np.searchsorted(count_bins, pivot_voice_usage_info['
pivot_voice_usage_info['call_count']
```

```
<ipython-input-130-2060bf1af780>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy)
  pivot_voice_usage_info['call_count'][col] = np.searchsorted(count_bins, pivot_voice_usage_info['c
all_count'][col].values)
```

Out[130]:

year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005	202006	202007	202008
event_source													
112053582	3	3	3	3	2	2	0	0	0	0	0	0	0
112053623	1	0	1	0	0	0	1	1	1	0	0	1	1
112053643	4	4	5	4	4	4	4	4	4	4	4	4	4
112054356	2	2	2	0	0	0	0	0	0	0	0	1	2
112055452	2	3	3	2	2	2	2	3	3	2	2	2	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
912286932	3	4	3	3	4	3	3	3	3	2	3	3	3
912286967	2	3	3	3	3	2	2	1	1	2	2	2	2
912286996	1	1	0	0	1	0	1	0	1	1	1	1	0
912287030	1	0	0	0	0	1	1	1	1	2	2	2	2
912287049	1	1	1	1	1	1	1	1	1	0	0	1	1

3329 rows × 13 columns

In [131]:
```
pivot_voice_usage_info['call_count'].describe()
```

Out[131]:

year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202
count	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.00000	3329.000
mean	2.031241	2.026735	2.072094	2.059778	2.111445	1.963953	2.015320	1.985581	1.68429	1.820
std	1.321370	1.305340	1.324090	1.295460	1.320790	1.256913	1.287213	1.263308	1.26567	1.259
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000
25%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000	1.000
50%	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	2.000
75%	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.00000	3.000
max	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000	7.000000	6.00000	7.000

In [132]:
```
# binning duration
duration_bins = [0, 840, 2640, 6420,12880,25720,60000,240000,709004]
for col in pivot_voice_usage_info['duration'].columns:
    pivot_voice_usage_info['duration'][col] = np.searchsorted(duration_bins, pivot_voice_usage_info[
pivot_voice_usage_info['duration']
```

```
<ipython-input-132-eb75694b315c>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  pivot_voice_usage_info['duration'][col] = np.searchsorted(duration_bins, pivot_voice_usage_info
['duration'][col].values)
```

Out[132]:

year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005	202006	202007	202008
event_source													
112053582	3	3	3	3	2	2	0	0	0	0	0	0	0
112053623	1	0	1	0	0	0	1	1	1	0	0	1	1
112053643	5	5	5	5	4	5	5	5	5	5	4	5	6
112054356	2	2	1	0	0	0	0	0	0	0	0	1	1
112055452	1	3	4	2	2	2	1	3	3	2	2	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
912286932	3	3	3	3	4	3	3	3	3	2	3	3	2
912286967	2	2	2	2	2	2	1	1	1	2	1	1	1
912286996	1	1	0	0	1	0	1	0	1	1	1	1	0
912287030	1	0	0	0	0	1	1	1	2	2	2	3	2
912287049	1	1	1	1	1	1	1	1	1	0	0	1	1

3329 rows × 13 columns

In [133]:
```
pivot_voice_usage_info['call_count'].describe()
```

Out[133]:

year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202
count	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.00000	3329.000
mean	2.031241	2.026735	2.072094	2.059778	2.111445	1.963953	2.015320	1.985581	1.68429	1.820
std	1.321370	1.305340	1.324090	1.295460	1.320790	1.256913	1.287213	1.263308	1.26567	1.259
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000
25%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000	1.000
50%	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	2.000
75%	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.00000	3.000
max	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000	7.000000	6.00000	7.000

In [134]:
```
year_month_list = pivot_voice_usage_info['call_count'].columns
```

In [135]:
```
# multplying bins to get usage rating
for col in year_month_list:
    pivot_voice_usage_info[col] = pivot_voice_usage_info['call_count'][col]  * pivot_voice_usage_inf
```

Usage ratings for year-month

In [136]:
```
pivot_voice_usage_info = pivot_voice_usage_info.drop(['call_count','duration'], axis=1)
pivot_voice_usage_info
```

Out[136]:

year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005	202006	202007	202008
event_source													
112053582	9	9	9	9	4	4	0	0	0	0	0	0	0
112053623	1	0	1	0	0	0	1	1	1	0	0	1	1
112053643	20	20	25	20	16	20	20	20	20	20	16	20	24
112054356	4	4	2	0	0	0	0	0	0	0	0	1	2
112055452	2	9	12	4	4	4	2	9	9	4	4	4	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
912286932	9	12	9	9	16	9	9	9	9	4	9	9	6
912286967	4	6	6	6	6	4	2	1	1	4	2	2	2
912286996	1	1	0	0	1	0	1	0	1	1	1	1	0
912287030	1	0	0	0	0	1	1	1	2	4	4	6	4
912287049	1	1	1	1	1	1	1	1	1	0	0	1	1

3329 rows × 13 columns

In [137]:
```
pivot_voice_usage_info.T.reset_index(drop=True).T
pivot_voice_usage_info.columns=year_month_list
```

In [138]:
```
# scaling usage rating
scaler = MinMaxScaler()
for col in year_month_list:
    pivot_voice_usage_info[col] = scaler.fit_transform(pivot_voice_usage_info[col].values.reshape(-1
pivot_voice_usage_info
```

Out[138]:

year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005	202006	202007
event_source												
112053582	0.140625	0.140625	0.140625	0.140625	0.062500	0.062500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
112053623	0.015625	0.000000	0.015625	0.000000	0.000000	0.000000	0.015625	0.017857	0.023810	0.000000	0.000000	0.015625
112053643	0.312500	0.312500	0.390625	0.312500	0.250000	0.312500	0.312500	0.357143	0.476190	0.408163	0.285714	0.312500
112054356	0.062500	0.062500	0.031250	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.015625
112055452	0.031250	0.140625	0.187500	0.062500	0.062500	0.062500	0.031250	0.160714	0.214286	0.081633	0.071429	0.062500
...	...	...	...	...	...	...	...	...	...	...	...	...
912286932	0.140625	0.187500	0.140625	0.140625	0.250000	0.140625	0.140625	0.160714	0.214286	0.081633	0.160714	0.140625
912286967	0.062500	0.093750	0.093750	0.093750	0.093750	0.062500	0.031250	0.017857	0.023810	0.081633	0.035714	0.031250
912286996	0.015625	0.015625	0.000000	0.000000	0.015625	0.000000	0.015625	0.000000	0.023810	0.020408	0.017857	0.015625
912287030	0.015625	0.000000	0.000000	0.000000	0.000000	0.015625	0.015625	0.017857	0.047619	0.081633	0.071429	0.093750
912287049	0.015625	0.015625	0.015625	0.015625	0.015625	0.015625	0.015625	0.017857	0.023810	0.000000	0.000000	0.015625

3329 rows × 13 columns

In [139]:
```
sns.heatmap(pivot_voice_usage_info.head(), annot=True)
```

Out[139]: <AxesSubplot:xlabel='year-month', ylabel='event_source'>

In [140]:
```
pivot_voice_usage_info['Usage_Rating_Scaled'] = pivot_voice_usage_info.sum(axis=1)/len(year_month_li
pivot_voice_usage_info
```

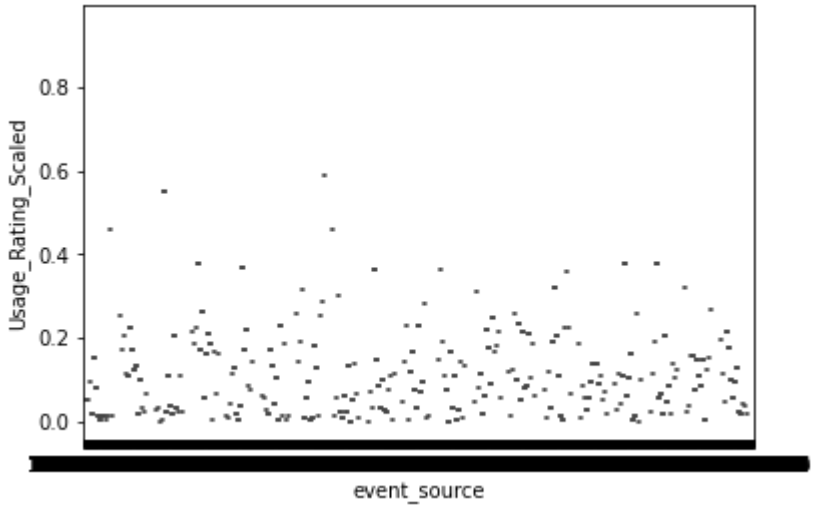
Out[140]:

year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005	202006	202007
event_source												
112053582	0.140625	0.140625	0.140625	0.140625	0.062500	0.062500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
112053623	0.015625	0.000000	0.015625	0.000000	0.000000	0.000000	0.015625	0.017857	0.023810	0.000000	0.000000	0.015625
112053643	0.312500	0.312500	0.390625	0.312500	0.250000	0.312500	0.312500	0.357143	0.476190	0.408163	0.285714	0.312500
112054356	0.062500	0.062500	0.031250	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.015625
112055452	0.031250	0.140625	0.187500	0.062500	0.062500	0.062500	0.031250	0.160714	0.214286	0.081633	0.071429	0.062500
...	...	...	...	...	...	...	...	...	...	...	...	...
912286932	0.140625	0.187500	0.140625	0.140625	0.250000	0.140625	0.140625	0.160714	0.214286	0.081633	0.160714	0.140625
912286967	0.062500	0.093750	0.093750	0.093750	0.093750	0.062500	0.031250	0.017857	0.023810	0.081633	0.035714	0.031250
912286996	0.015625	0.015625	0.000000	0.000000	0.015625	0.000000	0.015625	0.000000	0.023810	0.020408	0.017857	0.015625
912287030	0.015625	0.000000	0.000000	0.000000	0.000000	0.015625	0.015625	0.017857	0.047619	0.081633	0.071429	0.093750
912287049	0.015625	0.015625	0.015625	0.015625	0.015625	0.015625	0.015625	0.017857	0.023810	0.000000	0.000000	0.015625

3329 rows × 14 columns

Finding Outliers

In [159]:
```
sns.boxplot(x=pivot_voice_usage_info.index,y=pivot_voice_usage_info['Usage_Rating_Scaled'])
plt.show()
```



In [142]:
```
pivot_voice_usage_info.columns.get_level_values(level=0)[0]
```

Out[142]: 201908

In [156]:
```
for col in year_month_list:
    sns.boxplot(x=pivot_voice_usage_info[col].sample(n=10),y=pivot_voice_usage_info['Usage_Rating_Sc
    plt.show()
```

In [53]:
```
pivot_voice_usage_info.describe()
```

Out[53]:

year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005
count	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.000000	3329.00
mean	0.099312	0.097538	0.101124	0.099396	0.104193	0.091244	0.096289	0.111702	0.128295	0.11
std	0.102543	0.099801	0.103081	0.100766	0.103895	0.094929	0.099686	0.114240	0.147237	0.12
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.015625	0.015625	0.015625	0.015625	0.015625	0.015625	0.015625	0.017857	0.023810	0.02
50%	0.062500	0.062500	0.062500	0.062500	0.093750	0.062500	0.062500	0.071429	0.095238	0.08
75%	0.140625	0.140625	0.140625	0.140625	0.140625	0.140625	0.140625	0.160714	0.214286	0.18
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

In [54]:
```
# labling usage bins
bins = [0, 0.1, 0.3,1]
labels = ['LOW','MEDIUM','HIGH']
pivot_voice_usage_info['Usage_Rating_Categorical'] = pd.cut(pivot_voice_usage_info['Usage_Rating_Sca
pivot_voice_usage_info
```

Out[54]:

1910	201911	201912	202001	202002	202003	202004	202005	202006	202007	202008	Usage_Rating_Scaled	Usag
140625	0.140625	0.062500	0.062500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.052885	
015625	0.000000	0.000000	0.000000	0.015625	0.017857	0.023810	0.000000	0.000000	0.015625	0.015625	0.009215	
390625	0.312500	0.250000	0.312500	0.312500	0.357143	0.476190	0.408163	0.285714	0.312500	0.375000	0.339834	
031250	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.015625	0.031250	0.015625	
187500	0.062500	0.062500	0.062500	0.031250	0.160714	0.214286	0.081633	0.071429	0.062500	0.031250	0.092303	
...	...	...	...	...	...	...	...	...	...	...	...	
140625	0.140625	0.250000	0.140625	0.140625	0.160714	0.214286	0.081633	0.160714	0.140625	0.093750	0.153257	
093750	0.093750	0.093750	0.062500	0.031250	0.017857	0.023810	0.081633	0.035714	0.031250	0.031250	0.057905	
000000	0.000000	0.015625	0.000000	0.015625	0.000000	0.023810	0.020408	0.017857	0.015625	0.000000	0.010785	
000000	0.000000	0.000000	0.015625	0.015625	0.017857	0.047619	0.081633	0.071429	0.093750	0.062500	0.032436	
015625	0.015625	0.015625	0.015625	0.015625	0.017857	0.023810	0.000000	0.000000	0.015625	0.015625	0.014022	

In [55]:
```
# adding category "NO"
pivot_voice_usage_info['Usage_Rating_Categorical'] = pivot_voice_usage_info['Usage_Rating_Categorica
pivot_voice_usage_info.loc[pivot_voice_usage_info['Usage_Rating_Scaled'] == 0.0, 'Usage_Rating_Categ
# pivot_voice_usage_info[(pivot_voice_usage_info['Usage_Rating_Scaled'] == 0.0)]
```

Out[55]:

	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005	202006	202007	202008	Usage_Ra
year-month														
event_source														

Fina Result: Usage Ratings

In [56]:
```
pivot_voice_usage_info
```

Out[56]:

year-month	201908	201909	201910	201911	201912	202001	202002	202003	202004	202005	202006	202007
event_source												
112053582	0.140625	0.140625	0.140625	0.140625	0.062500	0.062500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
112053623	0.015625	0.000000	0.015625	0.000000	0.000000	0.000000	0.015625	0.017857	0.023810	0.000000	0.000000	0.015625
112053643	0.312500	0.312500	0.390625	0.312500	0.250000	0.312500	0.312500	0.357143	0.476190	0.408163	0.285714	0.312500
112054356	0.062500	0.062500	0.031250	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.015625
112055452	0.031250	0.140625	0.187500	0.062500	0.062500	0.062500	0.031250	0.160714	0.214286	0.081633	0.071429	0.062500
...	...	...	...	...	...	...	...	...	...	...	...	...
912286932	0.140625	0.187500	0.140625	0.140625	0.250000	0.140625	0.140625	0.160714	0.214286	0.081633	0.160714	0.140625
912286967	0.062500	0.093750	0.093750	0.093750	0.093750	0.062500	0.031250	0.017857	0.023810	0.081633	0.035714	0.031250
912286996	0.015625	0.015625	0.000000	0.000000	0.015625	0.000000	0.015625	0.000000	0.023810	0.020408	0.017857	0.015625
912287030	0.015625	0.000000	0.000000	0.000000	0.000000	0.015625	0.015625	0.017857	0.047619	0.081633	0.071429	0.093750
912287049	0.015625	0.015625	0.015625	0.015625	0.015625	0.015625	0.015625	0.017857	0.023810	0.000000	0.000000	0.015625

3329 rows × 15 columns

In [ ]:
```
```

Phase 2

Binnig by Location

- 1. group by location columns
- 2. Bin by location and get counts

In [57]:
```
# bucketing by location
voice_location_info = voice_usage_info
voice_location_info['Location_Identifier'] = voice_location_info['msan'].astype(str)+ voice_location
voice_location_info
```

Out[57]:

	event_source	call_count	duration	year-month	rtom_code	location_code	msan	equp_id	equp_index	Location_Identifier
0	412244863	2	186	202001	R-MH	MH-POL-NODE	MSAG5200-ISL	49145	1	MSAG5200-ISLMH-POL-NODE491451
1	412244863	3	114	202002	R-MH	MH-POL-NODE	MSAG5200-ISL	49145	1	MSAG5200-ISLMH-POL-NODE491451
2	662227788	10	2435	201909	R-MT	MT-AVR-NODE	MSAG5200-ISL	46716	1	MSAG5200-ISLMT-AVR-NODE467161
3	412244863	14	1061	202003	R-MH	MH-POL-NODE	MSAG5200-ISL	49145	1	MSAG5200-ISLMH-POL-NODE491451
4	412228632	65	7971	201910	R-MH	MH-POL-NODE	MSAG5200-ISL	49145	1	MSAG5200-ISLMH-POL-NODE491451
...	...	...	...	...	...	...	...	...	...	...
36151	112411111	182	20134	202008	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1	MSAG5200-ISLIDH-KGW-NODE6194491
36152	112411111	173	14409	201908	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1	MSAG5200-ISLIDH-KGW-NODE6194491
36153	112411111	196	19571	201910	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1	MSAG5200-ISLIDH-KGW-NODE6194491
36154	112411111	205	17685	201911	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1	MSAG5200-ISLIDH-KGW-NODE6194491
36155	112411111	242	19340	201912	R-KON	IDH-KGW-NODE	MSAG5200-ISL	619449	1	MSAG5200-ISLIDH-KGW-NODE6194491

36156 rows × 10 columns

In [58]:
```
voice_location_info["User_Count"] = 1
voice_gropued_by_locations = voice_location_info.groupby(['Location_Identifier'], axis=0,as_index=Fa
voice_gropued_by_locations = voice_gropued_by_locations[['Location_Identifier','User_Count']]
```

In [59]:
```
voice_gropued_by_locations
```

Out[59]:

	Location_Identifier	User_Count
0	C300MHE-DMD-NODE15221561	142
1	C300MHT-NE-NODE8152461	179
2	C300MIDH-IDJ-NODE11065541	699
3	C300MKI-NODE10668164	984
4	C300MNW-MEE-NODE18887751	1891
5	C300MPDT-NODE13379431	142
6	C300MTP-NODE8700911	234
7	C300MVH-NODE9874041	152
8	MA5603TAD-SRV-NODE10224991	89
9	MSAG5200-ISLBZ-NODE435881	1891
10	MSAG5200-ISLGE-NODE5577981	143
11	MSAG5200-ISLHC-NAP-NODE478721	192
12	MSAG5200-ISLHK-OCC-NODE506561	197
13	MSAG5200-ISLHNT-NODE423851	169
14	MSAG5200-ISLHO-NODE524692	39
15	MSAG5200-ISLHZ-BRL-NODE471911	184
16	MSAG5200-ISLIDH-KGW-NODE6194491	2139
17	MSAG5200-ISLJA-PKV-NODE460421	204
18	MSAG5200-ISLKI-KRB-NODE479411	9
19	MSAG5200-ISLMB-PSL-NODE482261	198
20	MSAG5200-ISLMH-POL-NODE491451	1330
21	MSAG5200-ISLMLT-VMD-NODE4145091	581
22	MSAG5200-ISLMT-AVR-NODE467161	3487
23	MSAG5200-ISLMV-GDM-NODE482491	3803
24	MSAG5200-ISLMX-NODE5391821	247

	Location_Identifier	User_Count
25	MSAG5200-ISLND-PHG-NODE500241	3845
26	MSAG5200-ISLNL-HGW-NODE491281	166
27	MSAG5200-ISLPC-AGT-NODE480031	1346
28	MSAG5200-ISLPK-PNU-NODE6136021	76
29	MSAG5200-ISLPRN-NODE427901	103
30	MSAG5200-ISLSI-DWS-NODE520721	921
31	MSAG5200-ISLTBL-NODE5577771	176
32	MSAG5200-ISLTBT-TLJ-NODE487241	97
33	MSAG5200-ISLVA-NKM-NODE481961	485
34	MSAG5200-ISLWI-NWT-NODE485201	226
35	MSAG5200-ISLWKN-SWP-NODE473481	370
36	MSAG5200-ISLWM-NODE432881	63
37	MSAG5200-ISLWY-MDB-NODE506221	197
38	MSAG5200BG-NODE431391	2568
39	MSAG5200HPG-NODE432141	209
40	MSAG5200MGE-NODE428381	192
41	MSAG5200PH-AKG-NODE431631	2967
42	MSAG5200PK-NODE438641	148
43	MSAG5200WLI-NODE422581	15
44	UA5000(IPMB)IM-NODE391531	196
45	UA5000(IPMB)SL-NODE385341	1753
46	ZXDSL9806H-ISLAG-PNP-NODE529761	227
47	ZXDSL9806H-ISLKE-NRP-NODE495731	232
48	ZXDSL9806H-ISLKL-PRC-NODE482361	253

In [ ]: