In [22]:
```python
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', 500)

#import or 'read' the csv file of the dataset
df_dpiData = pd.read_csv('device_performance_index.csv')
df_dpiData
```

Out[22]:

	Client_ID	Office_ID	BSOD_count	Hard_reset_count	Boot_Speed	Logon_Duration	CPU_Usage	Memory_Usage	System_Free_Space
0	1	88	0	0	9824	8520	0.005	0.000	180269625344
1	1	124	0	11	204075	34036	0.001	0.000	190256279552
2	1	172	1	21	51930	15774	0.000	0.000	198199214080
3	1	732	0	0	167701	3633	0.000	0.000	198328692736
4	1	172	0	1	235541	19340	0.009	0.000	177966186496
...	...	...	...	...	...	...	...	...	...
65164	9	283	0	0	8136	18113	0.002	0.114	21484277760
65165	9	636	0	0	30709	70357	0.000	0.000	377559977984
65166	9	626	0	0	10907	40337	0.020	0.000	156750188544
65167	9	124	0	0	12038	23904	0.004	0.000	145551847424
65168	9	183	0	0	32162	40562	0.004	0.000	383183306752

65169 rows × 9 columns

In [23]:
```python
#function to calculate the Xmin and Xmax takinkg into account the 2 and 98 percentiles
def xMin_xMax_calculator (dataFrame):

    xMinxMax = dataFrame.loc[:, ~dataFrame.columns.isin(['Client_ID', 'Office_ID'])].quantile([0.02, 0.98])

    return xMinxMax

resultado = xMin_xMax_calculator(df_dpiData)
resultado
```

Out[23]:

	BSOD_count	Hard_reset_count	Boot_Speed	Logon_Duration	CPU_Usage	Memory_Usage	System_Free_Space
0.02	0.0	0.0	4624.00	3708.36	0.000	0.00000	1.714007e+09
0.98	0.0	2.0	109884.68	96496.88	0.046	0.89164	4.471692e+11

In [24]:
```python
#make a copy of the original data set, to perfrom operations with the data

dfCopy = df_dpiData.copy()
#iterate through the columns and normalize each value according to the given formula
for column in dfCopy.columns[2:]:
    x = ((dfCopy[column]-resultado[column][0.02])/(resultado[column][0.98]-resultado[column][0.02]))
    #because both Xmin and Xmax (2 and 98 percentiles) for BSOD_count are 0, meaning that values greater than cero
    #are actually considered outliers, I replaced inf and nan values from Xnorm calculation.
    with pd.option_context('mode.use_inf_as_na', True):
        x.fillna(0, inplace=True)
    x = x.apply(lambda v: min(1, max(0, v)))
    dfCopy[column] = x

dfCopy
```

Out[24]:

	Client_ID	Office_ID	BSOD_count	Hard_reset_count	Boot_Speed	Logon_Duration	CPU_Usage	Memory_Usage	System_Free_Space
0	1	88	0	0.0	0.049401	0.051856	0.108696	0.000000	0.400839
1	1	124	0	1.0	1.000000	0.326847	0.021739	0.000000	0.423258
2	1	172	0	1.0	0.449418	0.130034	0.000000	0.000000	0.441089
3	1	732	0	0.0	1.000000	0.000000	0.000000	0.000000	0.441379
4	1	172	0	0.5	1.000000	0.168465	0.195652	0.000000	0.395668
...	...	...	...	...	...	...	...	...	...
65164	9	283	0	0.0	0.033365	0.155242	0.043478	0.127854	0.044382
65165	9	636	0	0.0	0.247813	0.718285	0.000000	0.000000	0.843735
65166	9	626	0	0.0	0.059690	0.394754	0.434783	0.000000	0.348040
65167	9	124	0	0.0	0.070435	0.217652	0.086957	0.000000	0.322901
65168	9	183	0	0.0	0.261617	0.397179	0.086957	0.000000	0.856358

65169 rows × 9 columns

In [25]:
```python
#iterate through the columns and flip the data over (1-v), except for the column System_Free_Space
for column in dfCopy.columns[2:8]:
    dfCopy[column] = dfCopy[column].apply(lambda v: 1-v)
dfCopy
```

Out[25]:

	Client_ID	Office_ID	BSOD_count	Hard_reset_count	Boot_Speed	Logon_Duration	CPU_Usage	Memory_Usage	System_Free_Space
0	1	88	1	1.0	0.950599	0.948144	0.891304	1.000000	0.400839
1	1	124	1	0.0	0.000000	0.673153	0.978261	1.000000	0.423258
2	1	172	1	0.0	0.550582	0.869966	1.000000	1.000000	0.441089
3	1	732	1	1.0	0.000000	1.000000	1.000000	1.000000	0.441379
4	1	172	1	0.5	0.000000	0.831535	0.804348	1.000000	0.395668
...	...	...	...	...	...	...	...	...	...
65164	9	283	1	1.0	0.966635	0.844758	0.956522	0.872146	0.044382
65165	9	636	1	1.0	0.752187	0.281715	1.000000	1.000000	0.843735
65166	9	626	1	1.0	0.940310	0.605246	0.565217	1.000000	0.348040
65167	9	124	1	1.0	0.929565	0.782348	0.913043	1.000000	0.322901
65168	9	183	1	1.0	0.738383	0.602821	0.913043	1.000000	0.856358

65169 rows × 9 columns

In [26]:
```python
#created a list with the names of the columns just to then, sum the values for each row, excluding the Client_Id and Office_id
col_list = list(dfCopy)

del col_list[0:2]

dfCopy['DPI'] = (10/7) * dfCopy[col_list].sum(axis=1)


dfCopy
```

Out[26]:

	Client_ID	Office_ID	BSOD_count	Hard_reset_count	Boot_Speed	Logon_Duration	CPU_Usage	Memory_Usage	System_Free_Space	DPI
0	1	88	1	1.0	0.950599	0.948144	0.891304	1.000000	0.400839	8.844123
1	1	124	1	0.0	0.000000	0.673153	0.978261	1.000000	0.423258	5.820959
2	1	172	1	0.0	0.550582	0.869966	1.000000	1.000000	0.441089	6.945196
3	1	732	1	1.0	0.000000	1.000000	1.000000	1.000000	0.441379	7.773399
4	1	172	1	0.5	0.000000	0.831535	0.804348	1.000000	0.395668	6.473643
...	...	...	...	...	...	...	...	...	...	...
65164	9	283	1	1.0	0.966635	0.844758	0.956522	0.872146	0.044382	8.120633
65165	9	636	1	1.0	0.752187	0.281715	1.000000	1.000000	0.843735	8.396623
65166	9	626	1	1.0	0.940310	0.605246	0.565217	1.000000	0.348040	7.798305
65167	9	124	1	1.0	0.929565	0.782348	0.913043	1.000000	0.322901	8.496939
65168	9	183	1	1.0	0.738383	0.602821	0.913043	1.000000	0.856358	8.729437

65169 rows × 10 columns

In [27]:
```python
# I add the calculated DPI to the original dataFrame just to show the raw data and the calculated DPI
df_dpiData['DPI'] = dfCopy['DPI']
df_dpiData
```

Out[27]:

	Client_ID	Office_ID	BSOD_count	Hard_reset_count	Boot_Speed	Logon_Duration	CPU_Usage	Memory_Usage	System_Free_Space	DPI
0	1	88	0	0	9824	8520	0.005	0.000	180269625344	8.844123
1	1	124	0	11	204075	34036	0.001	0.000	190256279552	5.820959
2	1	172	1	21	51930	15774	0.000	0.000	198199214080	6.945196
3	1	732	0	0	167701	3633	0.000	0.000	198328692736	7.773399
4	1	172	0	1	235541	19340	0.009	0.000	177966186496	6.473643
...	...	...	...	...	...	...	...	...	...	...
65164	9	283	0	0	8136	18113	0.002	0.114	21484277760	8.120633
65165	9	636	0	0	30709	70357	0.000	0.000	377559977984	8.396623
65166	9	626	0	0	10907	40337	0.020	0.000	156750188544	7.798305
65167	9	124	0	0	12038	23904	0.004	0.000	145551847424	8.496939
65168	9	183	0	0	32162	40562	0.004	0.000	383183306752	8.729437

65169 rows × 10 columns

In [28]:
```python
#I group the data by client ID and generate some statistics based on the calculated DPI

#first I create quantile functions to aggregate later
def q25(x):
    return x.quantile(0.25)
def q75(x):
    return x.quantile(0.75)
client_group = df_dpiData.groupby(['Client_ID']).DPI.agg(['count', 'mean', 'std', 'median', 'min', 'max', q25, q75])
client_group
```

Out[28]:

Client_ID	count	mean	std	median	min	max	q25	q75
1	747	8.171885	0.749689	8.383374	5.258783	9.869012	7.715706	8.705456
2	27061	7.579661	0.958438	7.754950	2.381598	9.736532	7.013942	8.343630
3	77	8.616359	0.908401	8.514863	6.312591	9.956028	8.007548	9.384459
4	10849	8.414967	0.603874	8.541943	4.811278	9.985133	8.145786	8.812709
5	1431	8.542190	0.684110	8.735744	4.703036	9.962908	8.207064	9.002747
6	1360	8.080344	0.877923	8.316343	4.370388	9.761934	7.471366	8.895051
7	11870	7.916748	0.784605	8.013568	3.393475	9.940652	7.419211	8.455225
8	9965	8.756074	0.784118	8.851732	4.547216	10.000000	8.373187	9.369694
9	1809	7.908599	0.830721	8.170680	4.120810	9.635803	7.408426	8.429599

Info between customers

- The client with the best DPI is number 8 with mean value of 8.76
- The client with the worse DPI is number 2 with a mean value of 7.58. This has the greatest std and the lowest min value detected. It has the greatest number of devices tested
- Client 3 has the lowest number of devices tested

In [29]:
```python
#Data was grouped by Client Id and by Office Id to evaluate differences between offices for each client.
agglist = ['count', 'mean', 'std', 'median', 'min', 'max', q25, q75]
clientAndOffice_group = df_dpiData.groupby(['Client_ID', 'Office_ID'])['DPI'].agg(agglist)
clientAndOffice_group.head(100)

#NaN values for std exist because there are some offices that have only one device producing data, hence the std
#cannot be calculated.
```

Out[29]:

Client_ID	Office_ID	count	mean	std	median	min	max	q25	q75
1	27	86	8.454018	0.610713	8.680782	5.810839	9.411342	8.232742	8.844941
	31	86	8.166130	0.641990	8.383439	5.800290	9.064540	7.683414	8.624845
	88	41	8.227088	0.556931	8.375909	6.417684	8.940462	7.878805	8.619285
	124	58	7.878709	0.771962	8.195738	5.820959	8.800981	7.352118	8.400421
	172	46	7.591013	0.918005	7.729383	5.258783	9.396145	6.894910	8.349101
	341	66	8.460337	0.576154	8.578089	6.716137	9.869012	8.278858	8.859189
	388	16	7.933274	0.628050	8.135633	6.891943	8.620400	7.303096	8.506286
	476	108	8.110853	0.733442	8.352471	6.286455	9.734616	7.530620	8.653282
	520	1	8.794619	NaN	8.794619	8.794619	8.794619	8.794619	8.794619
	628	79	8.434985	0.761667	8.657372	6.047212	9.801078	8.079202	8.820428
	640	61	8.084380	0.778487	8.221568	6.468547	9.721114	7.756424	8.642450
	674	17	7.497695	0.933376	7.554477	5.685907	8.718273	6.652990	8.314833
	729	38	8.177435	0.753214	8.497729	6.552737	9.224278	7.767326	8.708321
	732	44	8.268340	0.645235	8.375478	7.004057	9.649851	7.864907	8.636122
2	3	1	8.376967	NaN	8.376967	8.376967	8.376967	8.376967	8.376967
	17	21	6.844332	1.212932	6.823554	4.011074	8.397450	5.823197	8.118114
	33	293	8.051635	0.738483	8.215316	4.806223	9.079818	7.711240	8.595093
	55	169	7.730112	0.652156	7.868073	5.027486	8.722924	7.428945	8.237283
	59	3	8.146420	0.378652	8.362205	7.709203	8.367854	8.035704	8.365029
	64	6	8.298537	0.058785	8.287147	8.240704	8.397582	8.255473	8.322082
	82	4	8.192672	0.375711	8.332581	7.641258	8.464271	8.116982	8.408271
	86	27	7.395845	0.685538	7.110297	6.211036	8.567395	6.916950	8.058976
	87	64	7.786243	0.543026	7.929310	6.323400	8.588378	7.429864	8.171194
	101	14	7.786371	0.474997	7.819990	6.888521	8.402859	7.490604	8.186800
	105	18	7.738280	0.854346	7.845307	4.980837	8.732039	7.632594	8.287436
	106	25	7.734301	0.710647	7.984259	6.490549	8.862247	7.084031	8.415583
	118	641	8.105900	0.614716	8.321586	5.254111	8.998036	7.792870	8.490040
	121	278	7.545329	0.713935	7.618562	5.382568	9.029983	7.059167	8.117552
	143	285	7.918150	0.700932	8.165915	5.113418	9.362767	7.618477	8.407809
	146	60	8.225964	0.591254	8.419722	6.067302	8.991016	7.868946	8.581664
	148	1	8.454323	NaN	8.454323	8.454323	8.454323	8.454323	8.454323
	162	28	7.959108	0.686907	8.162197	5.948082	8.835236	7.528179	8.482712
	164	211	8.273088	0.797016	8.573317	5.301286	9.003197	7.990655	8.845167
	173	184	8.171587	0.636567	8.279510	5.688965	9.390363	7.751595	8.711494
	179	15	7.967021	0.506359	8.243370	6.836474	8.406072	7.670899	8.319590
	197	87	7.515669	0.731409	7.627188	4.730053	8.967483	7.104579	7.974628
	202	17	8.027746	0.623276	8.269629	6.977963	8.860756	7.343712	8.414171
	206	164	7.793288	0.762034	7.947203	4.961012	8.867513	7.464562	8.341072
	209	21	8.103250	0.417721	8.092633	7.105415	8.874228	7.882987	8.391610
	210	15	7.877343	0.596423	8.049858	6.659847	8.940818	7.494750	8.246172
	218	3	8.463824	0.437795	8.574815	7.981217	8.835442	8.278016	8.705128
	234	1	8.733729	NaN	8.733729	8.733729	8.733729	8.733729	8.733729
	241	160	8.058130	0.741379	8.332739	5.056013	8.998062	7.627461	8.532763
	242	4	7.752270	0.741030	7.901567	6.782946	8.422999	7.376964	8.276873
	247	6	8.291151	0.388391	8.331101	7.855693	8.812445	7.946082	8.531893
	257	1	6.878254	NaN	6.878254	6.878254	6.878254	6.878254	6.878254
	259	52	7.937004	0.682636	8.263401	5.966914	8.650444	7.411285	8.529323
	260	1	7.299251	NaN	7.299251	7.299251	7.299251	7.299251	7.299251
	268	28	8.202936	0.285814	8.323303	7.547874	8.613164	8.101911	8.376880
	277	114	7.468650	0.757354	7.537437	5.220304	9.000174	6.990869	8.155533
	284	48	7.768097	0.739140	8.005566	5.451472	8.629515	7.308720	8.332376
	288	1	6.105795	NaN	6.105795	6.105795	6.105795	6.105795	6.105795
	291	1	9.004440	NaN	9.004440	9.004440	9.004440	9.004440	9.004440
	294	114	7.472510	0.859550	7.507245	4.716356	8.986500	6.830062	8.148703
	299	3172	8.025167	0.805040	8.197076	3.603380	9.736532	7.593636	8.704383
	306	14	7.397994	0.664383	7.311512	6.239849	8.485285	6.882221	7.703555
	308	7	7.842303	0.655787	7.948934	6.502701	8.368313	7.722314	8.315772
	331	24	7.638905	0.594081	7.564711	6.557779	8.637152	7.120329	8.179949
	340	1	6.493293	NaN	6.493293	6.493293	6.493293	6.493293	6.493293
	342	21	8.194408	0.830740	8.127454	6.408649	9.559534	7.752853	8.728573
	343	241	8.111588	0.671909	8.311265	5.163914	9.015136	7.750239	8.615142
	346	48	7.523474	0.849486	7.596636	4.096931	8.887167	7.180988	8.007788
	348	375	8.203202	0.617892	8.387567	5.598019	8.985769	7.887544	8.613320
	375	1	8.010025	NaN	8.010025	8.010025	8.010025	8.010025	8.010025
	376	3	7.732024	0.661829	7.609824	7.139811	8.446437	7.374818	8.028131
	379	11	7.942970	0.650729	7.822976	6.787022	8.857614	7.547554	8.369284
	387	68	7.855019	0.804080	8.092716	5.544217	8.923062	7.367058	8.417210
	408	357	7.679268	0.799409	7.833995	4.737521	8.946700	7.141255	8.296400
	409	1032	8.145759	0.630668	8.348826	4.630298	9.024791	7.817408	8.577107
	412	1653	6.666183	0.850572	6.792694	3.110545	8.338149	6.206569	7.281312
	419	1132	7.776083	0.815173	7.935710	3.755632	9.006340	7.377253	8.370150
	433	1	8.655496	NaN	8.655496	8.655496	8.655496	8.655496	8.655496
	439	110	7.916521	0.660210	8.068089	6.205486	9.038964	7.352972	8.470215
	446	556	7.894914	0.635115	7.991653	5.101319	8.909982	7.517094	8.439218
	448	6	7.560045	0.498674	7.420203	7.002549	8.278539	7.238941	7.900241
	468	2577	6.660426	0.890648	6.819455	3.075605	8.463587	6.095810	7.314185
	492	5	6.871858	0.931613	6.907885	5.606816	8.218796	6.656720	6.969072
	493	319	7.722792	0.962866	7.983949	4.458751	8.763078	7.206955	8.491511
	523	41	7.783203	0.790386	7.870099	4.476937	8.586631	7.571793	8.369942
	536	1	8.547717	NaN	8.547717	8.547717	8.547717	8.547717	8.547717
	537	3	7.255955	1.271759	7.515450	5.874461	8.377953	6.694955	7.946701
	555	2629	7.860062	0.724610	7.932958	4.456308	9.331206	7.500148	8.412508
	557	2498	6.665593	0.872246	6.813617	2.381598	8.430720	6.165816	7.292198
	560	434	7.872734	0.564609	8.043408	5.426331	8.953368	7.569767	8.247413
	571	5	7.726051	0.537421	7.642933	7.183392	8.519864	7.318273	7.965791
	579	13	7.625886	0.709629	7.482435	6.489115	8.930181	7.244755	7.993050
	586	3	7.192658	0.829155	7.588537	6.239762	7.749673	6.914150	7.669105
	590	3	8.033478	0.446072	7.855062	7.704229	8.541142	7.779646	8.198102
	609	2	8.413573	0.172747	8.413573	8.291422	8.535724	8.352497	8.474648
	616	10	7.396885	0.914766	7.430684	5.738154	8.806416	6.900829	7.982231
	618	1523	7.673836	0.799320	7.883386	3.521026	9.130858	7.237637	8.288670
	624	59	7.592510	0.911839	7.777483	3.744053	8.812178	7.173549	8.220488
	632	2717	7.838746	0.782121	7.930921	3.588561	9.400224	7.387007	8.429436
	637	1	8.991791	NaN	8.991791	8.991791	8.991791	8.991791	8.991791
	643	1	7.654541	NaN	7.654541	7.654541	7.654541	7.654541	7.654541
	657	20	7.696840	0.582040	7.626821	6.736371	8.571642	7.306371	8.245339
	663	8	7.549039	0.611547	7.572487	6.522071	8.607212	7.242178	7.746713
	664	14	7.807901	0.480163	7.860518	6.838187	8.472616	7.706476	8.132753
	675	15	7.501276	1.154795	7.860979	4.647226	9.252968	6.910469	8.237576
	678	3	7.592522	0.281530	7.508130	7.362840	7.906596	7.435485	7.707363

In [30]:
```python
#I filter the data by client and then group by Office ID to check statistics for the offices of each client
filter = df_dpiData['Client_ID'] == 9
clientx = df_dpiData.loc[filter].groupby(['Office_ID'])['DPI'].agg(agglist)
clientx.sort_values(['mean'], ascending=False)
```

Out[30]:

Office_ID	count	mean	std	median	min	max	q25	q75
325	125	8.338611	0.759651	8.382025	5.480292	9.635803	7.972079	8.923352
636	88	8.287706	0.847565	8.505908	5.791823	9.423953	7.741597	8.898824
286	88	8.256066	0.721475	8.339709	5.532641	9.201489	7.933593	8.782814
183	75	8.137348	0.751959	8.330968	6.461862	9.297146	7.694659	8.692346
165	143	8.074530	0.568321	8.268443	6.476816	9.296101	7.696607	8.357664
558	66	8.005608	0.642340	8.239494	6.081578	9.053716	7.591810	8.375990
283	77	8.002829	0.610789	8.166592	6.559551	9.202488	7.588644	8.354198
548	89	7.971950	0.861818	8.187225	5.589966	9.293341	7.326482	8.573202
662	63	7.945899	0.527472	8.166391	6.549155	9.004939	7.600366	8.302399
580	59	7.942674	0.771226	8.170680	5.568677	9.137082	7.553790	8.423963
16	58	7.939659	0.768567	8.271674	5.389393	9.182808	7.674420	8.357785
639	60	7.926060	0.698953	8.203318	5.738146	9.172661	7.578834	8.378671
89	65	7.922837	0.671449	7.977611	6.599469	8.817831	7.455707	8.477086
66	97	7.921181	0.817367	8.040576	5.651571	9.173312	7.330527	8.578027
543	106	7.882102	0.957829	8.222527	4.293282	9.177579	7.548278	8.372324
368	14	7.831024	0.859468	8.285238	5.840448	8.809411	7.421332	8.385416
124	135	7.805232	0.883163	7.989730	4.374913	9.011255	7.334937	8.495698
608	53	7.706973	1.033916	7.906441	4.609292	9.202889	7.078711	8.433584
501	54	7.686102	0.819939	7.838197	5.620358	8.773499	7.094816	8.368835
724	53	7.620340	0.718622	7.929845	5.325344	8.390189	6.967732	8.226369
626	81	7.476957	0.816922	7.613190	5.029285	8.682074	6.794582	8.149066
690	61	7.427465	0.864313	7.512163	5.774874	8.957954	6.738340	8.247695
469	51	7.350593	0.827344	7.399996	5.673347	8.619353	6.823634	7.925779
730	39	7.121525	0.871138	7.110321	4.549963	8.554370	6.508748	7.614737
221	9	7.027217	1.574970	7.574435	4.120810	8.690864	5.692523	7.986194

Info between offices for different clients

Client 1

- office 520 has the best DPI, however there's only one device in that specific office. Office 341 follows with a mean DPI of 8.46

Client 2

- office 291 and 637 have the highest mean DPI, but only one device in each of them
- office 704 and 573 have the highest std which means that the info provided by the divices in those offices is quite disperse
- there are 8 offices which mean DPI values are below 7

Client 3

- It has only one office, with 77 devices
- it's mean DPI is above 8.5

Client 4

- It has no office with a mean DPI lower than 7.5

Client 5

- It has no office with a mean DPI lower than 7.5
- office 287 has the highest mean DPI with 9.24

Client 6

- office 23 and 499 have the lowest mean DPIs with 6.97 and 5.99 respectively.

Client 7

- has several offices with a mean DPI below 7
- the lowest min value registered was for office 431 with a mean DPI of 3.39

Client 8

- No office with a mean DPI below 7.5
- various offices with mean DPI values over 9

Client 9

- No office with a mean DPI below 7
- highest mean DPI for office 325 with 8.33

In [31]:
```python
#Created a filter to see all the offices that have a mean DPI below 7
filter = clientAndOffice_group['mean'] <7
clientAndOffice_group[filter]
```

Out[31]:

Client_ID	Office_ID	count	mean	std	median	min	max	q25	q75
2	17	21	6.844332	1.212932	6.823554	4.011074	8.397450	5.823197	8.118114
	257	1	6.878254	NaN	6.878254	6.878254	6.878254	6.878254	6.878254
	288	1	6.105795	NaN	6.105795	6.105795	6.105795	6.105795	6.105795
	340	1	6.493293	NaN	6.493293	6.493293	6.493293	6.493293	6.493293
	412	1653	6.666183	0.850572	6.792694	3.110545	8.338149	6.206569	7.281312
	468	2577	6.660426	0.890648	6.819455	3.075605	8.463587	6.095810	7.314185
	492	5	6.871858	0.931613	6.907885	5.606816	8.218796	6.656720	6.969072
	557	2498	6.665593	0.872246	6.813617	2.381598	8.430720	6.165816	7.292198
6	23	4	6.973558	0.970273	6.548319	6.374490	8.423103	6.477978	7.043899
	499	5	5.997368	0.996315	5.746265	5.106860	7.420532	5.132200	6.580984
7	38	6	6.869442	0.626077	6.859928	6.083954	7.714024	6.421708	7.282369
	44	2	6.521125	0.100426	6.521125	6.450112	6.592137	6.485619	6.556631
	61	1	6.721726	NaN	6.721726	6.721726	6.721726	6.721726	6.721726
	73	1	5.761661	NaN	5.761661	5.761661	5.761661	5.761661	5.761661
	99	1	6.911381	NaN	6.911381	6.911381	6.911381	6.911381	6.911381
	145	5	6.854340	0.695116	7.093210	5.959120	7.521621	6.288096	7.409653
	158	1	6.326429	NaN	6.326429	6.326429	6.326429	6.326429	6.326429
	240	2	6.292511	0.602436	6.292511	5.866524	6.718497	6.079517	6.505504
	301	3	6.175305	1.639125	7.109535	4.282658	7.133723	5.696096	7.121629
	318	1	6.758238	NaN	6.758238	6.758238	6.758238	6.758238	6.758238
	402	2	6.214709	0.202859	6.214709	6.071265	6.358152	6.142987	6.286430
	487	2	6.889915	0.248040	6.889915	6.714524	7.065306	6.802220	6.977611
	502	1	5.811328	NaN	5.811328	5.811328	5.811328	5.811328	5.811328
	530	4	6.583282	1.376408	6.990798	4.591943	7.759591	6.366086	7.207995
	551	1	6.945235	NaN	6.945235	6.945235	6.945235	6.945235	6.945235
	583	1	6.575030	NaN	6.575030	6.575030	6.575030	6.575030	6.575030
	604	3	6.773804	1.401647	7.237710	5.199015	7.884686	6.218362	7.561198
	644	4	6.253084	1.011877	6.549904	4.830034	7.082496	5.893387	6.909602
	677	21	6.692853	0.749772	6.528320	5.350307	8.256411	6.283381	7.191265
	713	3	6.469681	0.153351	6.437352	6.335072	6.636620	6.386212	6.536986
	739	2	6.490284	0.487941	6.490284	6.145258	6.835310	6.317771	6.662797