In [55]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt

import random
train_file = "data/train.csv"
test_file = "data/test.csv"
random.seed(300)
```

In [96]:
```
#Ran once to get the total number of records
#n = sum(1 for line in open(train_file)) - 1

#print(n)
#n = 37670293
```

In [97]:
```
# Get first 1000 rows to get basic info
df = pd.read_csv(train_file, nrows=1000)
```

In [98]:
```
### feature information
print(df.columns)
print(len(df.columns))
```

```
Index(['date_time', 'site_name', 'posa_continent', 'user_location_country',
       'user_location_region', 'user_location_city',
       'orig_destination_distance', 'user_id', 'is_mobile', 'is_package',
       'channel', 'srch_ci', 'srch_co', 'srch_adults_cnt', 'srch_children_cnt',
       'srch_rm_cnt', 'srch_destination_id', 'srch_destination_type_id',
       'is_booking', 'cnt', 'hotel_continent', 'hotel_country', 'hotel_market',
       'hotel_cluster'],
      dtype='object')
24
```

In [99]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 24 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   date_time                   1000 non-null   object 
 1   site_name                   1000 non-null   int64  
 2   posa_continent              1000 non-null   int64  
 3   user_location_country       1000 non-null   int64  
 4   user_location_region        1000 non-null   int64  
 5   user_location_city          1000 non-null   int64  
 6   orig_destination_distance   268 non-null    float64
 7   user_id                     1000 non-null   int64  
 8   is_mobile                   1000 non-null   int64  
 9   is_package                  1000 non-null   int64  
 10  channel                     1000 non-null   int64  
 11  srch_ci                     1000 non-null   object 
 12  srch_co                     1000 non-null   object 
 13  srch_adults_cnt             1000 non-null   int64  
 14  srch_children_cnt           1000 non-null   int64  
 15  srch_rm_cnt                 1000 non-null   int64  
 16  srch_destination_id         1000 non-null   int64  
 17  srch_destination_type_id    1000 non-null   int64  
 18  is_booking                  1000 non-null   int64  
 19  cnt                         1000 non-null   int64  
 20  hotel_continent             1000 non-null   int64  
 21  hotel_country               1000 non-null   int64  
 22  hotel_market                1000 non-null   int64  
 23  hotel_cluster               1000 non-null   int64  
dtypes: float64(1), int64(20), object(3)
memory usage: 175.8+ KB
```

In [100…
```
df.head()
```

Out[100…

	date_time	site_name	posa_continent	user_location_country	user_location_region	user_location_city	orig_destination_distance	user_id
0	2014-08-11 07:46:59	2	3	66	348	48862	2234.2641	12
1	2014-08-11 08:22:12	2	3	66	348	48862	2234.2641	12
2	2014-08-11 08:24:33	2	3	66	348	48862	2234.2641	12
3	2014-08-09 18:05:16	2	3	66	442	35390	913.1932	93
4	2014-08-09 18:08:18	2	3	66	442	35390	913.6259	93

5 rows × 24 columns

In [101…
```
df.describe()
```

Out[101…

	site_name	posa_continent	user_location_country	user_location_region	user_location_city	orig_destination_distance	user_id
count	1000.00000	1000.00000	1000.000000	1000.000000	1000.000000	268.000000	1000.000000
mean	19.36100	2.16700	50.865000	193.805000	19680.638000	1860.755094	3596.333000
std	10.30577	0.74274	56.595334	243.919765	16541.209223	2271.610410	1499.094642
min	2.00000	0.00000	3.000000	12.000000	1493.000000	3.337900	12.000000
25%	13.00000	2.00000	23.000000	48.000000	4924.000000	177.330075	2451.000000
50%	24.00000	2.00000	23.000000	64.000000	10067.000000	766.156100	3972.000000
75%	25.00000	3.00000	66.000000	189.000000	40365.000000	2454.858800	4539.000000
max	37.00000	4.00000	205.000000	991.000000	56440.000000	8457.263600	6450.000000

8 rows × 21 columns

Hotels are grouped together based on historical price, customer star ratings, geographical locations relative to city center, etc are represented by hotel cluster. So in the table below, srch_destination_id 8250 has 3 cluster 1 hotels, one of which was booked 8 8/11 at 8:22:12. checkin date from 8/29 to 9/2. Two searches resulted in click(not booked)

In [220…
```
df[['date_time','srch_destination_id','hotel_cluster','is_booking','srch_ci','srch_co']].head(20)
```

Out[220…

	date_time	srch_destination_id	hotel_cluster	is_booking	srch_ci	srch_co
0	2014-08-11 07:46:59	8250	1	False	2014-08-27	2014-08-31
1	2014-08-11 08:22:12	8250	1	True	2014-08-29	2014-09-02
2	2014-08-11 08:24:33	8250	1	False	2014-08-29	2014-09-02
3	2014-08-09 18:05:16	14984	80	False	2014-11-23	2014-11-28
4	2014-08-09 18:08:18	14984	21	False	2014-11-23	2014-11-28
5	2014-08-09 18:13:12	14984	92	False	2014-11-23	2014-11-28
6	2014-07-16 09:42:23	8267	41	False	2014-08-01	2014-08-02
7	2014-07-16 09:45:48	8267	41	False	2014-08-01	2014-08-02
8	2014-07-16 09:52:11	8267	69	False	2014-08-01	2014-08-02
9	2014-07-16 09:55:24	8267	70	False	2014-08-01	2014-08-02
10	2014-07-16 10:00:06	8267	98	False	2014-08-01	2014-08-02
11	2014-07-16 10:02:58	8267	10	False	2014-08-01	2014-08-02
12	2014-01-17 06:24:56	8291	18	False	2014-04-17	2014-04-20
13	2014-01-18 14:33:31	8291	28	False	2014-04-16	2014-04-19
14	2014-01-21 06:39:08	8291	25	False	2014-04-17	2014-04-20
15	2014-01-21 06:40:18	8291	25	False	2014-04-18	2014-04-20
16	2014-01-22 06:10:02	8291	25	False	2014-04-18	2014-04-20
17	2014-01-24 11:52:04	8291	25	False	2014-04-17	2014-04-19
18	2014-01-24 17:26:24	8291	25	False	2014-04-17	2014-04-19
19	2014-02-27 17:44:23	8291	2	False	2014-04-17	2014-04-19

Now read in 50,000 records

In [268…
```
# Set data types for better  memory management.
dtype = {'site_name':np.int32,
'posa_continent':np.int32,
'user_location_country':np.int32,
'user_location_region':np.int32,
'user_location_city':np.int32,
'orig_destination_distance':np.float32,
'user_id':np.int32,
'is_mobile':np.int32,
'is_package':np.int32,
'channel':np.int32,
'srch_adults_cnt':np.int32,
'srch_children_cnt':np.int32,
'srch_rm_cnt':np.int32,
'srch_destination_id':np.int32,
'srch_destination_type_id':np.int32,
'is_booking':bool,
'cnt':np.int32,
'hotel_continent':np.int32,
'hotel_country':np.int32,
'hotel_market':np.int32,
'hotel_cluster':np.int32}

# Set to datetime type for date calculations calculations
date_time','srch_ci', 'srch_co']
```

In [354…
```
df = pd.read_csv('data/train.csv',dtype=dtype,parse_dates = parse_dates,nrows=50000)
```

In [355…
```
total = df.isnull().sum().sort_values(ascending = False)
percent = (df.isnull().sum()/df.isnull().count()*100).sort_values(ascending = False)
pd.concat([total, percent], axis=1, keys=['Total', 'Percent']).transpose()
```

Out[355…

	orig_destination_distance	srch_ci	srch_co	channel	site_name	posa_continent	user_location_country	user_location_region	user_
Total	19002.000	29.000	29.000	0.0	0.0	0.0	0.0	0.0	
Percent	38.004	0.058	0.058	0.0	0.0	0.0	0.0	0.0	

2 rows × 24 columns

In [356…
```
df = df.dropna(subset=['srch_ci', 'srch_co'])
total = df.isnull().sum().sort_values(ascending = False)
percent = (df.isnull().sum()/df.isnull().count()*100).sort_values(ascending = False)
pd.concat([total, percent], axis=1, keys=['Total', 'Percent']).transpose()
```

Out[356…

	orig_destination_distance	hotel_cluster	hotel_market	site_name	posa_continent	user_location_country	user_location_region	use
Total	18997.000000	0.0	0.0	0.0	0.0	0.0	0.0	
Percent	38.016049	0.0	0.0	0.0	0.0	0.0	0.0	

2 rows × 24 columns

Compute number of nights in the search and add as new feature

In [357…
```
# get number of booked nights as difference between check in and check out
hotel_nights = df['srch_co'] - df['srch_ci']
hotel_nights = (hotel_nights / np.timedelta64(1, 'D')).astype(float) # convert to float to avoid NA problems
df['hotel_nights'] = hotel_nights
df[['date_time','srch_destination_id','hotel_cluster','is_booking','srch_ci','srch_co','hotel_nights']].head()
```

Out[357…

	date_time	srch_destination_id	hotel_cluster	is_booking	srch_ci	srch_co	hotel_nights
0	2014-08-11 07:46:59	8250	1	False	2014-08-27	2014-08-31	4.0
1	2014-08-11 08:22:12	8250	1	True	2014-08-29	2014-09-02	4.0
2	2014-08-11 08:24:33	8250	1	False	2014-08-29	2014-09-02	4.0
3	2014-08-09 18:05:16	14984	80	False	2014-11-23	2014-11-28	5.0
4	2014-08-09 18:08:18	14984	21	False	2014-11-23	2014-11-28	5.0

Get the total number of bookings per srch_destination_id, and hotel cluster. This will tell us the total number of bookings for each cluster in the dataset.

Clusters with most booking are the most popular. Now we have to find out more bout the search criteria. Seach destination id is a 1st criteria.

In [358…
```
df.groupby(['srch_destination_id','hotel_cluster'])['is_booking'].agg(['sum','count'])
```

Out[358…

		sum	count
srch_destination_id	**hotel_cluster**		
11	**94**	1	2
14	**20**	1	3
	75	0	2
16	**7**	1	2
19	**20**	0	1
...	**...**	...	...
64871	**46**	0	2
64999	**54**	0	1
65035	**10**	1	7
	35	0	1
	36	0	1

16077 rows × 2 columns

Table below shows search destination id 24, has 6 hotel clusters. Cluster 3 was searched once and no booking. Cluster 32 was search 4 times resulting in 2 bookings-both with one adult and no children. We can compute a measure for a hotel cluster that will rate it as popular or unpopular.

In [359…
```
df[df['srch_destination_id']==24][['srch_adults_cnt', 'srch_children_cnt','srch_rm_cnt','srch_destination_id','hotel_cluster'
```

Out[359…

	srch_adults_cnt	srch_children_cnt	srch_rm_cnt	srch_destination_id	hotel_cluster	is_booking

	srch_adults_cnt	srch_children_cnt	srch_rm_cnt	srch_destination_id	hotel_cluster	is_booking
34527	1	0	1	24	3	False
34528	1	0	1	24	32	False
34529	1	0	1	24	32	True
34530	1	0	1	24	76	False
34531	1	0	1	24	32	False
34532	1	0	1	24	94	True
34533	1	0	1	24	32	True
34550	1	0	1	24	42	False
40929	1	0	2	24	91	False

In [360…
```
df[df['srch_destination_id']==24][['srch_destination_id','hotel_cluster','is_booking']].groupby(['srch_destination_id','hotel_
```

Out[360…

		sum	count
srch_destination_id	**hotel_cluster**		
24	**3**	0	1
	32	2	4
	42	0	1
	76	0	1
	91	0	1
	94	1	1

Compute bookings and clicks. Bookings are aggregated with those with value of 1, clicks are the total number of searches in each group

In [361…
```
df_aggs = df.groupby(['srch_destination_id','hotel_cluster'])['is_booking'].agg(['sum','count']).reset_index()
df_aggs[df_aggs['srch_destination_id']==24]
```

Out[361…

	srch_destination_id	hotel_cluster	sum	count
26	24	3	0	1
27	24	32	2	4
28	24	42	0	1
29	24	76	0	1
30	24	91	0	1
31	24	94	1	1

Compute the relevance of hotel cluster based on number of bookings per search. relevance is high when number of bookings is closer to number of search results. This is weighted by total number of searches as shown below and number of bookings. Relevence of 0 is the lowest with no bookings at all(regardless of number of searches)

In [362…
```
df_agg  = df.groupby(['srch_destination_id','hotel_cluster'])['is_booking'].agg(['sum','count']).reset_index()
df_agg = df_agg.rename(columns={'sum':'bookings','count':'clicks'})
df_agg['relevance'] = df_agg['bookings'] + 0.05*(df_agg['clicks']* df_agg['bookings'])
```

The tables below show that the relevance of hotel clusters varies depending on the srch_destinationID. For example hotel cluster 3 at destination of 24 has relevance of 0 but 1.05 for destination 235.

In [363…
```
# Show relevance per search destination id.
df_agg[(df_agg['srch_destination_id'] == 24) | (df_agg['srch_destination_id'] == 235)]
```

Out[363…

	srch_destination_id	hotel_cluster	bookings	clicks	relevance
26	24	3	0	1	0.00
27	24	32	2	4	2.40
28	24	42	0	1	0.00
29	24	76	0	1	0.00
30	24	91	0	1	0.00
31	24	94	1	1	1.05
120	235	3	1	1	1.05
121	235	77	0	1	0.00

This table shows variance of relevance for each srch_destination_id. This can be surmized to be true for other search criterial. We will look at srch_adult_cnt below.

In [403…
```
df_agg[df_agg['hotel_cluster'] == 3][['hotel_cluster','relevance']]
```

Out[403…

	hotel_cluster	relevance

	hotel_cluster	relevance
3	3	48.60
103	3	96.30
203	3	1.45
303	3	0.00
403	3	0.00
497	3	0.00

In [365…
```
df_agg.describe()
```

Out[365…

	srch_destination_id	hotel_cluster	bookings	clicks	relevance
count	16077.000000	16077.000000	16077.000000	16077.000000	16077.000000
mean	17593.380979	48.731977	0.255272	3.108229	0.468041
std	12770.400526	28.897645	0.724159	7.527323	5.346835
min	11.000000	0.000000	0.000000	1.000000	0.000000
25%	8740.000000	23.000000	0.000000	1.000000	0.000000
50%	12602.000000	47.000000	0.000000	2.000000	0.000000
75%	23956.000000	75.000000	0.000000	3.000000	0.000000
max	65035.000000	99.000000	34.000000	300.000000	544.000000

In [366…
```
df_agg[df_agg['relevance'] >= 544]
```

Out[366…

	srch_destination_id	hotel_cluster	bookings	clicks	relevance
2717	8250	1	34	300	544.0

In [367…
```
def most_popular_in_group(group, n_max=10):
    relevance = group['relevance'].values
    hotel_cluster = group['hotel_cluster'].values
    most_popular = hotel_cluster[np.argsort(relevance)[::-1]][:n_max]
    return np.array_str(most_popular)[1:-1] # remove square brackets
```

In [368…
```
most_pop = df_agg.groupby(['srch_destination_id']).apply(most_popular_in_group)
most_pop = pd.DataFrame(most_pop).rename(columns={0:'hotel_cluster'})
most_pop.head(10)
```

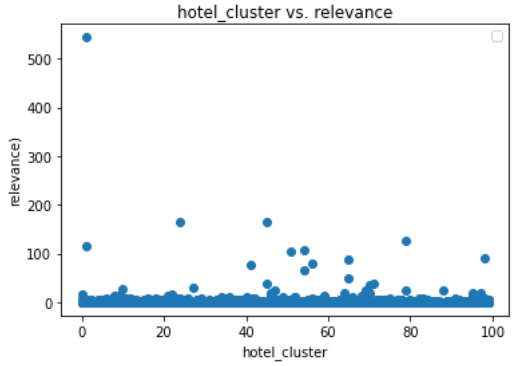
Out[368…

	hotel_cluster
srch_destination_id	
11	94
14	20 75
16	7
19	61 40 30 20
21	44 46 82 11 15 20 29 30 36 85
24	32 94 91 76 42 3
25	48 90
27	89 86 30
33	78
40	82

The graph below shows a few outliers. These are the ones with more clicks.

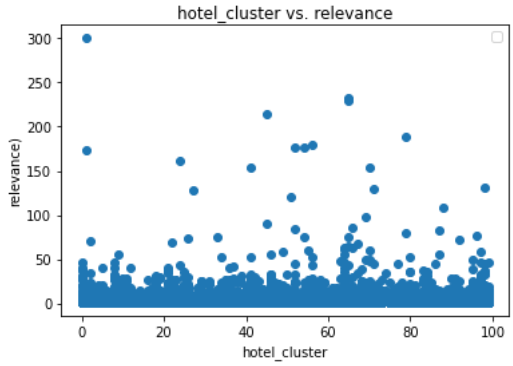
In [372…
```
plt.scatter(df_agg['hotel_cluster'],df_agg['relevance'])
plt.title('hotel_cluster vs. relevance')
plt.xlabel('hotel_cluster')
plt.ylabel('relevance)')
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



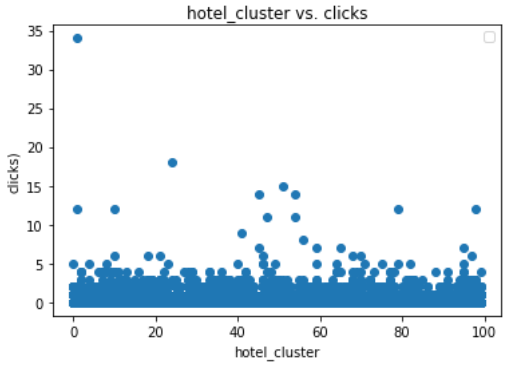
In [373…
```
plt.scatter(df_agg['hotel_cluster'],df_agg['clicks'])
plt.title('hotel_cluster vs. clicks')
plt.xlabel('hotel_cluster')
plt.ylabel('clicks)')
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



In [375…
```
plt.scatter(df_agg['hotel_cluster'],df_agg['bookings'])
plt.title('hotel_cluster vs. bookings')
plt.xlabel('hotel_cluster')
plt.ylabel('bookings)')
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



In [331…
```
df_agg
```

Out[331…

	srch_destination_id	hotel_cluster	bookings	clicks	relevance
0	11	94	1	2	1.10
1	14	20	1	3	1.15
2	14	75	0	2	0.00
3	16	7	1	2	1.10
4	19	20	0	1	0.00
...	...	...	...	...	...
16076	64871	46	0	2	0.00
16077	64999	54	0	1	0.00
16078	65035	10	1	7	1.35
	65035	35	0	1	0.00

	srch_destination_id	hotel_cluster	bookings	clicks	relevance
16080	65035	36	0	1	0.00

16081 rows × 5 columns

This shows which clusters have the highest relevance. Cluster 1 showed earlier to have a relevance 0f 544 with 300 clicks and 34 bookings

In [332…
```
most_pop_all = df_agg.groupby('hotel_cluster')['relevance'].sum().nlargest(5).index
most_pop_all = np.array_str(most_pop_all)[1:-1]
most_pop_all
```

Out[332…
```
' 1 91 45 54 41'
```

In [333…
```
most_pop_all
```

Out[333…
```
' 1 91 45 54 41'
```

Compute relevance based on srch_adults_cnt

In [381…
```
df_agg  = df[df['srch_adults_cnt'] >0].groupby(['srch_adults_cnt','hotel_cluster'])['is_booking'].agg(['sum','count']).reset_
df_agg = df_agg.rename(columns={'sum':'bookings','count':'clicks'})
df_agg['relevance'] = df_agg['bookings'] + 0.05*(df_agg['clicks']* df_agg['bookings'])
```

In [382…
```
# Show relevance per search destination id.
df_agg[df_agg['relevance'] > 0 ]
```

Out[382…

	srch_adults_cnt	hotel_cluster	bookings	clicks	relevance
0	1	0	7	91	38.85
1	1	1	15	86	79.50
2	1	2	25	148	210.00
3	1	3	9	88	48.60
4	1	4	13	112	85.80
…	…	…	…	…	…
624	8	47	1	2	1.10
627	8	51	1	3	1.15
631	8	61	1	7	1.35
633	8	63	1	2	1.10
648	8	95	1	5	1.25

397 rows × 5 columns

In [383…
```
most_pop = df_agg.groupby(['srch_adults_cnt']).apply(most_popular_in_group)
most_pop = pd.DataFrame(most_pop).rename(columns={0:'hotel_cluster'})
most_pop.head(10)
```

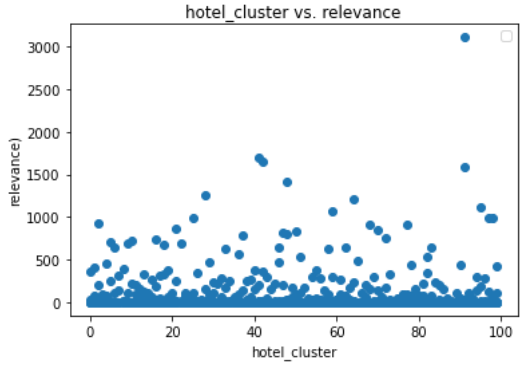
Out[383…

	hotel_cluster
srch_adults_cnt	
1	91 48 46 42 82 18 59 5 21 64
2	91 41 42 48 28 64 95 59 25 97
3	91 48 62 16 42 41 5 50 95 25
4	16 1 91 48 2 25 42 72 8 41
5	25 58 19 98 42 91 41 30 50 43
6	83 51 50 91 36 25 1 47 6 54
7	28 81 95 70 61 51 45 19 16 9
8	61 95 42 13 51 47 63 30 43 41
9	98 91 78 76 70 59 56 50 48 41

This graph shows that relevance with respect to adult count is different than those with respect to search destination.

In [387…
```
plt.scatter(df_agg['hotel_cluster'],df_agg['relevance'])
plt.title('hotel_cluster vs. relevance')
plt.xlabel('hotel_cluster')
plt.ylabel('relevance)')
plt.legend()
plt.show()
```

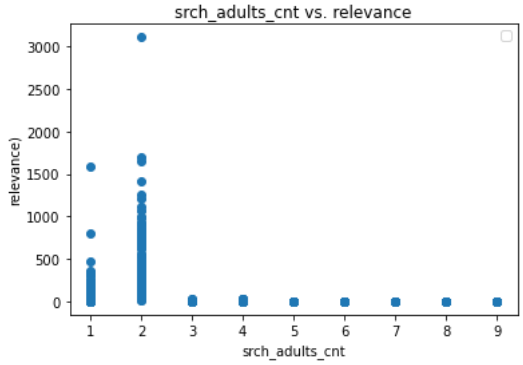
```
No handles with labels found to put in legend.
```



Adult count 1 and 2 seem to have most relevance

In [388…
```
plt.scatter(df_agg['srch_adults_cnt'],df_agg['relevance'])
plt.title('srch_adults_cnt vs. relevance')
plt.xlabel('srch_adults_cnt')
plt.ylabel('relevance)')
plt.legend()
plt.show()
```

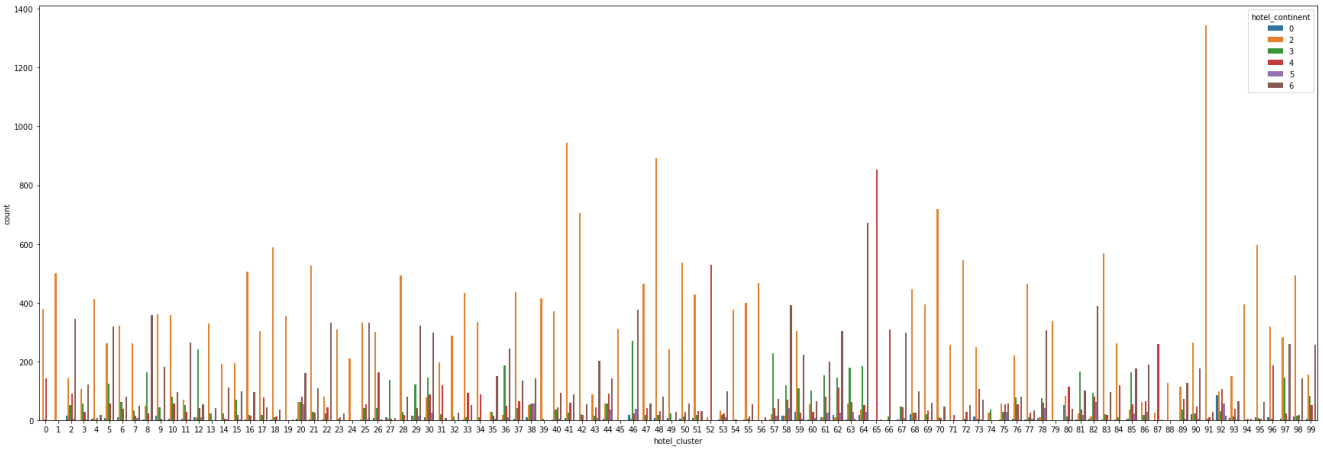
```
No handles with labels found to put in legend.
```



Distribuition of hotel clusters in continents. Continent 2 has the most clustets.

In [392…
```
fig_dims =(30, 10)
fig, ax = plt.subplots(figsize=fig_dims)
sns.countplot(x='hotel_cluster', hue='hotel_continent', ax=ax, data=df)
```

Out[392… <AxesSubplot:xlabel='hotel_cluster', ylabel='count'>

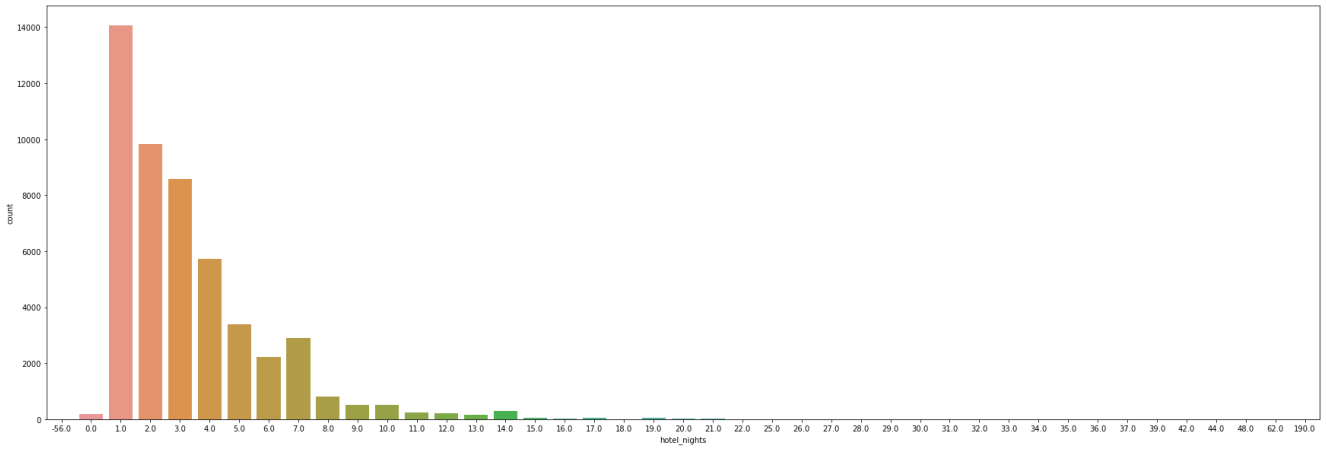


In [394…
```
plt.figure(figsize=(11, 9))
sns.countplot(x="hotel_nights", data=train)
```

Out[394… <AxesSubplot:xlabel='hotel_cluster', ylabel='count'>

In [396…
```
fig_dims =(30, 10)
fig, ax = plt.subplots(figsize=fig_dims)
sns.countplot(x='hotel_nights', ax=ax, data=df)
```

Out[396… <AxesSubplot:xlabel='hotel_nights', ylabel='count'>



Based on this priliminary analysis, the dataset can be augmented with data to better caculate the relevance of each cluster. Using relevance as target variable and classifying it as binary popular/unpopular", or nominnally as 'low_popular/med_popular/high_popular", would allow us to run logistic regression, KNN, or kmeans clustering algorithms. Keeping relevance as continuous will allow us to run linear regression model. Decsion tree can be used as a first pass to better undertsand the data.