In [1]:

```
# Put these at the top of every notebook, to get automatic reloading and inl
from IPython.core.display import display, HTML
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

%reload_ext autoreload
%autoreload 1
%matplotlib inline

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)

display(HTML("<style>.container { width:100% !important; }</style>"))
```

In [2]:

```
import os
import seaborn as sns
import pandas as pd
import math
import numpy as np

from Utils.UtilsGeoViz import *
from Utils.UtilsViz import *
from Utils.DataUtils import *
```

In [3]:

```
US_coord = [37.0902, -102]
NY_COORD = [40.7128, -74.0060]

# data_path = "C:\\Users\\sriharis\\OneDrive\\UChicago\\DataMining\\project\
data_path = os.path.join(os.getcwd(), "../data/listings.csv")
listings = pd.read_csv(data_path, index_col="id")
PERCENTILE_CROP = [1,98]
display(listings.shape)
```

```
(50228, 105)
```

In [4]:

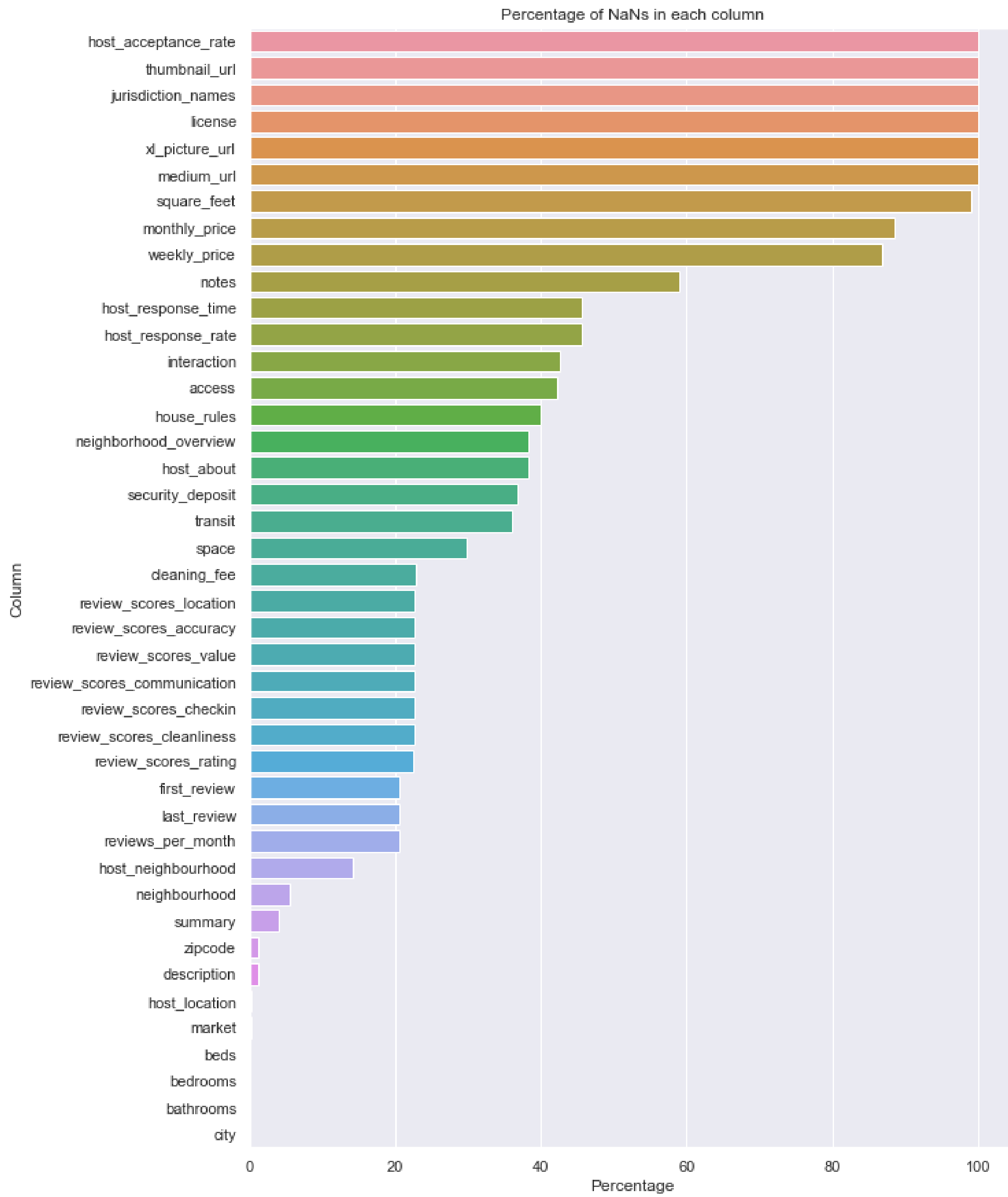
```
def remove_childlist_from_parentlist(parent, child):
    return [x for x in parent if x not in child]
```

In [5]:
```
def plot_nans(df, ax, annot=True, filter_out_zeros=True):
    nan_df = analyse_nans(df)
    if filter_out_zeros:
        cols_to_keep = []
        for col in nan_df.columns:
            if nan_df[col].iloc[1] > 0:
                cols_to_keep.append(col)
        nan_df = nan_df[cols_to_keep]
    nan_df_transpose = nan_df.T
    nan_df_transpose.sort_values(by="percentage", ascending=False, axis=0, i
    sns.barplot(data=nan_df_transpose, y=nan_df_transpose.index, x="percenta
    ax.set(ylabel="Column", xlabel="Percentage", title="Percentage of NaNs i
#     plot_bar(data=nan_df_transpose, x=nan_df_transpose.index, y="percentag
#              ax=ax, annot=annot, highlight_max_min=False,
#              xlabel="Column", ylabel="Percentage", title="Percentage of Na
#             xrot=90, plot_mean=False)
```

In [6]:
```
rows_to_drop = listings[listings['host_listings_count'] != listings['host_to
listings.drop(index=rows_to_drop,axis=0,inplace=True)
```

In [43]:

```
f, ax = plt.subplots(1,1, figsize=(10,15))
sns.set()
plot_nans(listings, ax=ax, annot=False, filter_out_zeros=True)
```

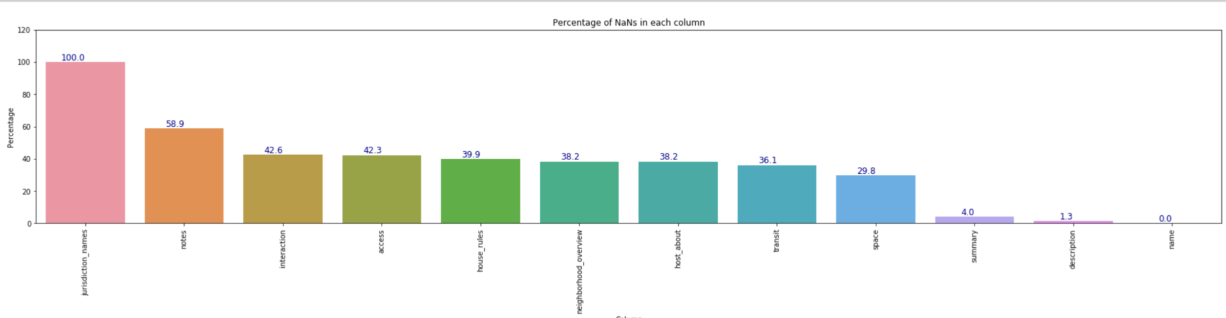


In [14]:

```
monu_cols_to_clean = ['id','host_response_time', 'host_response_rate',
        'host_is_superhost',
        'host_listings_count',
        'host_total_listings_count', 'host_verifications',
        'host_has_profile_pic', 'host_identity_verified',
        'is_location_exact', 'property_type', 'room_type', 'accommodates',
        'bathrooms', 'bedrooms', 'beds', 'bed_type', 'amenities', 'square_fee
        'price', 'weekly_price', 'monthly_price', 'security_deposit',
        'cleaning_fee', 'guests_included', 'extra_people','instant_bookable',
        'is_business_travel_ready', 'cancellation_policy',
        'require_guest_profile_picture', 'require_guest_phone_verification',
        'calculated_host_listings_count', 'reviews_per_month']

cols_to_drop = ['listing_url','scrape_id', 'last_scraped','experiences_offer
                'host_url', 'host_name', 'host_acceptance_rate', 'host_thumbn
                'license']

col_list_1 = find_unique_elems([listings.columns, monu_cols_to_clean])

nlp_cols = \
['jurisdiction_names',
 'notes',
 'interaction',
 'access',
 'house_rules',
 'neighborhood_overview',
 'host_about',
 'transit',
 'space',
 'summary',
 'name',
 'description']

ssh_cols_to_clean = remove_childlist_from_parentlist(col_list_1, cols_to_dro
ssh_cols_to_clean = remove_childlist_from_parentlist(ssh_cols_to_clean, nlp_
```

In [17]:

```
nlp_listings = listings[nlp_cols]
plot_nans(nlp_listings)
```



In [18]:
```
nlp_lencols = ["description", "host_about"]
nlp_listings["description"].fillna("", inplace=True)
nlp_listings["host_about"].fillna("", inplace=True)

nlp_listings["description"] = nlp_listings["description"].astype(str)
nlp_listings["host_about"] = nlp_listings["host_about"].astype(str)

nlp_listings["desc_len"] = nlp_listings["description"].apply(len)
nlp_listings["host_about_len"] = nlp_listings["host_about"].apply(len)
```

In [19]:
```
nlp_len_cols = ["description", "host_about", "desc_len", "host_about_len"]
nlp_listings[nlp_len_cols].head()
```

Out[19]:

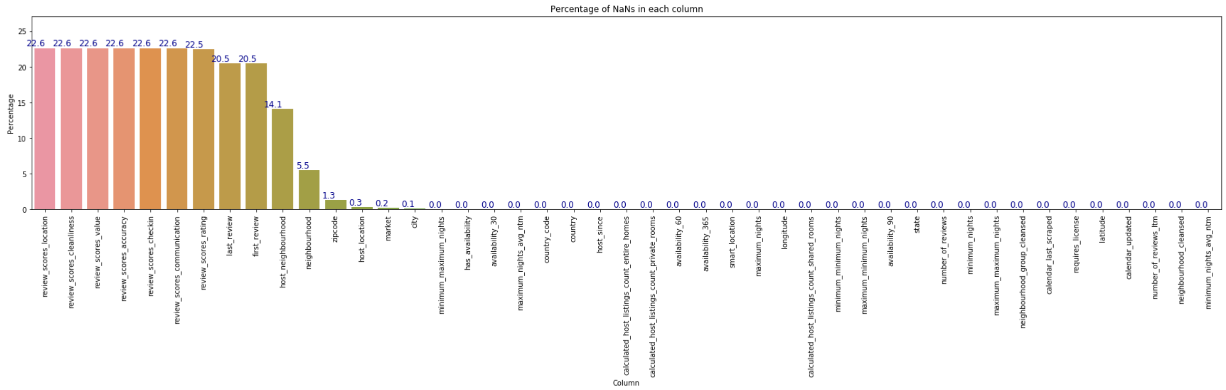
	description	host_about	desc_len	host_about_len
0	Great light, exposed brick and 10 feet high ce...		412	0
1	Renovated apt home in elevator building. Spaci...	Educated professional living in Brooklyn. I l...	392	431
2	Find your romantic getaway to this beautiful, ...	A New Yorker since 2000! My passion is creatin...	1000	427
3	This is a spacious, clean, furnished master be...	From Brooklyn with love.	1000	25
4	WELCOME TO OUR INTERNATIONAL URBAN COMMUNITY T...	Make Up Artist National/ (Website hidden by Ai...	1000	354

In [20]:
```
cols_to_drop
```

Out[20]:
```
['listing_url',
 'scrape_id',
 'last_scraped',
 'experiences_offered',
 'thumbnail_url',
 'medium_url',
 'picture_url',
 'xl_picture_url',
 'host_id',
 'host_url',
 'host_name',
 'host_acceptance_rate',
 'host_thumbnail_url',
 'host_picture_url',
 'street',
 'license']
```

In [21]:
```
listings = listings[ssh_cols_to_clean]
```

In [22]:
```
plot_nans(listings)
```



Make placeholders

In [23]:
```
more_cols_to_drop = []
```

In [24]:
```
listings.head()
```

Out[24]:

	availability_365	neighbourhood_cleansed	longitude	review_scores_value	calculated_host_listing
0	65	Midtown	-73.967679	NaN	
1	365	Kensington	-73.972370	10.0	
2	365	Midtown	-73.983774	9.0	
3	290	Williamsburg	-73.942362	10.0	
4	365	Harlem	-73.941902	NaN	

In [25]:
```
listings.shape
```

Out[25]: (50220, 45)

Country

In [26]:
```
listings["country"].value_counts()
```

Out[26]:
```
United States    50220
Name: country, dtype: int64
```

In [27]:
```
listings["Peru"==listings["country"]]
```

Out[27]:

availability_365	neighbourhood_cleansed	longitude	review_scores_value	calculated_host_listings_

Drop column? Peru seems to be an outlier. listings url is also invalid when checked.

Review scores

Hypothesis: Most rows in the **review_scores_** columns that have NaNs, have them for all the **review_scores_** columns in that particular row.

i.e., if *review_scores_value* has a NaN, then so will *review_scores_location* and the other *review_scores_* columns.

Let's verify this by finding the columns that are NOT NaNs throughout all the review_scores_* columns.

In [7]:
```

review_scores_cols = ['review_scores_value', 'review_scores_location', 'revi
nan_df = analyse_nans(listings[review_scores_cols])


# Verify that these columns have missing values in the exact same rows.
print("Number of rows between pairs of columns that have NaNs only in one of
for i in range(1, len(review_scores_cols)):
    print(review_scores_cols[i], "\t", review_scores_cols[i-1], end=" -->\t
    idx1 = nan_df[review_scores_cols[i-1]].iloc[2]
    idx2 = nan_df[review_scores_cols[i]].iloc[2]
    l = find_unique_elems([idx1, idx2])
    print(len(l), "rows, \ti.e.", round(100*len(l)/listings.shape[0], 2), "%
```

```
Number of rows between pairs of columns that have NaNs only in one of them -
review_scores_location   review_scores_value -->        17 rows,       i.e. 0.
03 % of total listings rows
review_scores_checkin    review_scores_location -->     19 rows,       i.e. 0.
04 % of total listings rows
review_scores_cleanliness        review_scores_checkin -->       43 rows,
i.e. 0.09 % of total listings rows
review_scores_communication      review_scores_cleanliness -->   28 rows,
i.e. 0.06 % of total listings rows
review_scores_accuracy   review_scores_communication -->         31 rows,
i.e. 0.06 % of total listings rows
review_scores_rating     review_scores_accuracy -->     47 rows,       i.e. 0.
09 % of total listings rows
```

Majority of rows have NaNs throughout all the review_scores_* columns. However, there are a few that don't as well.

Plot a distribution of Review Scores

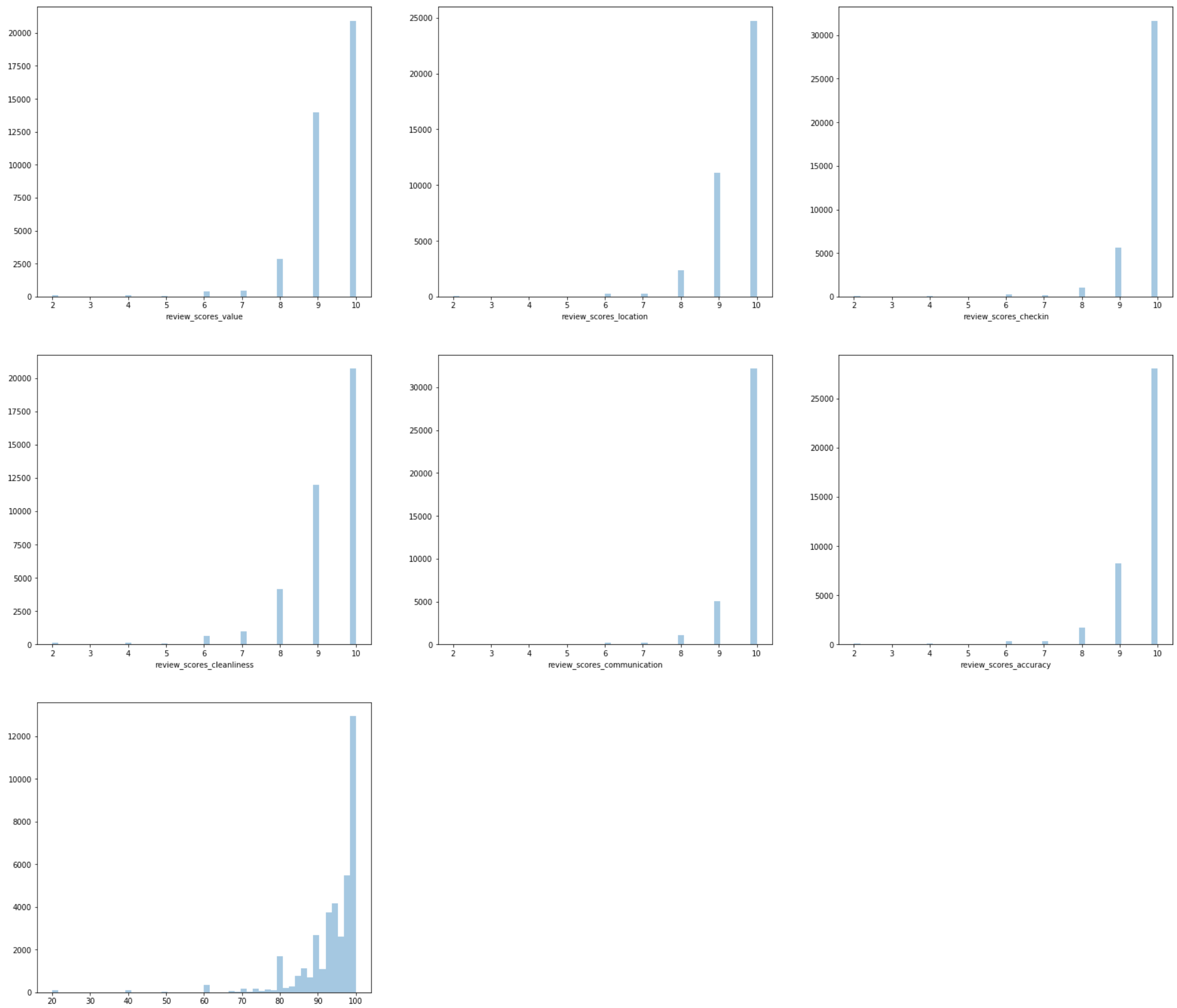
In [8]:
```
listings[review_scores_cols].describe()
```

Out[8]:

	review_scores_value	review_scores_location	review_scores_checkin	review_scores_cleanline
count	38848.000000	38847.000000	38852.000000	38889.0000
mean	9.376184	9.527016	9.738315	9.2653
std	0.902918	0.776485	0.711515	1.0786
min	2.000000	2.000000	2.000000	2.0000
25%	9.000000	9.000000	10.000000	9.0000
50%	10.000000	10.000000	10.000000	10.0000
75%	10.000000	10.000000	10.000000	10.0000
max	10.000000	10.000000	10.000000	10.0000

In [9]:

```python
# Uncomment these lines to see histograms of the review scores columns
f, ax = plt.subplots(3,3, figsize=(27,24))
col_counter = 0
g = sns.distplot(a=listings[~listings[review_scores_cols[col_counter]].isna(
col_counter+=1
g = sns.distplot(a=listings[~listings[review_scores_cols[col_counter]].isna(
col_counter+=1
g = sns.distplot(a=listings[~listings[review_scores_cols[col_counter]].isna(
col_counter+=1
g = sns.distplot(a=listings[~listings[review_scores_cols[col_counter]].isna(
col_counter+=1
g = sns.distplot(a=listings[~listings[review_scores_cols[col_counter]].isna(
col_counter+=1
g = sns.distplot(a=listings[~listings[review_scores_cols[col_counter]].isna(
col_counter+=1
g = sns.distplot(a=listings[~listings[review_scores_cols[col_counter]].isna(
col_counter+=1
ax[2][1].set_visible(not ax[2][1].get_visible())
ax[2][2].set_visible(not ax[2][2].get_visible())
```

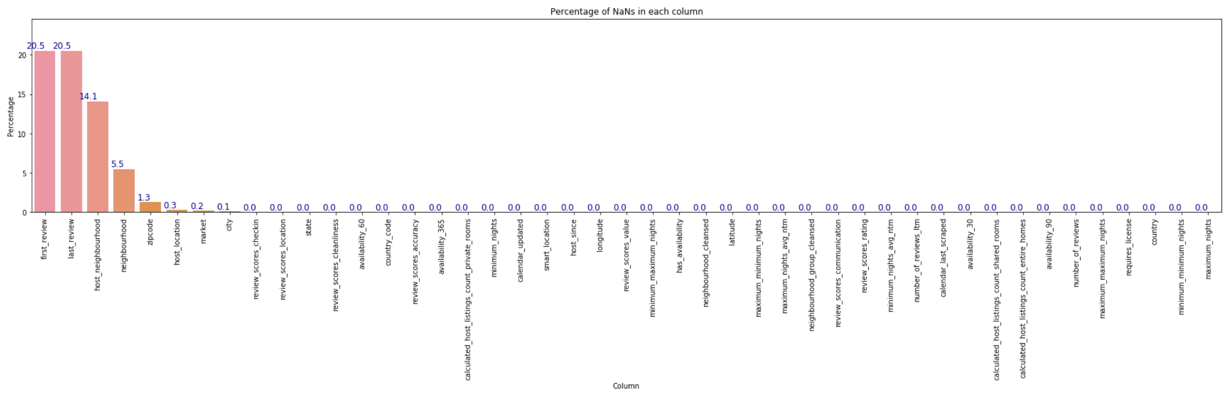


Well, skewed distributions throughout as expected. Transformations might prove useful here to compensate for the large left skew.

We could fill the missing values of each columns with the integer part of its median.

In [26]:
```
for col in review_scores_cols:
    listings[col].fillna(listings[col].median(), inplace=True)
```

In [27]:
```
plot_nans(listings)
```



city, market, zipcode, neighbourhood, host_neighbourhood

In [28]:
```
geo_cols = ["city", "neighbourhood","neighbourhood_cleansed", "neighbourhood
listings[listings["city"].isna()][geo_cols].head()
```

Out[28]:

	city	neighbourhood	neighbourhood_cleansed	neighbourhood_group_cleansed	market	hos
8208	NaN	Sunset Park	Sunset Park	Brooklyn	New York	
10646	NaN	Astoria	Astoria	Queens	New York	
10857	NaN	NaN	Astoria	Queens	New York	
10880	NaN	Astoria	Astoria	Queens	New York	
10882	NaN	NaN	Astoria	Queens	New York	

city

In [29]:
```
listings["city"].fillna("NYC", inplace=True)
```

market

In [30]:
```
listings["market"].value_counts()
```

Out[30]:
```
New York                      50066
Other (Domestic)                 19
Adirondacks                       2
Catskills and Hudson Valley       1
Los Angeles                       1
Cuba                              1
Agra                              1
Atlanta                           1
Paris                             1
San Francisco                     1
D.C.                              1
Boston                            1
Jamaica South Coast               1
South Bay, CA                     1
New Orleans                       1
Kyoto                             1
Lagos, NG                         1
Name: market, dtype: int64
```

In [31]:
```
no_geonan_listings = listings[geo_cols].dropna()
```

In [32]:
```
for market in no_geonan_listings["market"].unique():
    print(market, "  -->  ", no_geonan_listings[no_geonan_listings["market"]
```

```
New York   -->   ['Manhattan' 'Brooklyn' 'Queens' 'Staten Island' 'Bronx']
Other (Domestic)   -->   ['Queens' 'Brooklyn' 'Manhattan']
Boston   -->   ['Brooklyn']
Los Angeles   -->   ['Manhattan']
Atlanta   -->   ['Brooklyn']
Paris   -->   ['Brooklyn']
New Orleans   -->   ['Brooklyn']
San Francisco   -->   ['Manhattan']
Cuba   -->   ['Queens']
D.C.   -->   ['Brooklyn']
Lagos, NG   -->   ['Brooklyn']
Agra   -->   ['Manhattan']
Catskills and Hudson Valley   -->   ['Brooklyn']
Kyoto   -->   ['Brooklyn']
Adirondacks   -->   ['Brooklyn']
Jamaica South Coast   -->   ['Queens']
```

In [33]:
```
map_nbc_market = {}
for nbc in no_geonan_listings["neighbourhood_cleansed"].unique():
    map_nbc_market[nbc] = listings[listings["neighbourhood_cleansed"]==nbc][
```

In [34]:
```
map_nbc_market
```

Out[34]:
```
{'Midtown': array(['New York', nan], dtype=object),
 'Kensington': array(['New York', 'Other (Domestic)'], dtype=object),
 'Williamsburg': array(['New York', 'D.C.', 'Kyoto', nan], dtype=object),
 'Harlem': array(['New York', 'Agra', nan], dtype=object),
 'Clinton Hill': array(['New York', nan], dtype=object),
 "Hell's Kitchen": array(['New York', nan], dtype=object),
 'Upper West Side': array(['New York', nan], dtype=object),
 'Flatiron District': array(['New York', 'Other (Domestic)', nan], dtype=obje
ct),
 'Chinatown': array(['New York'], dtype=object),
 'Upper East Side': array(['New York', nan, 'Other (Domestic)'], dtype=objec
t),
 'South Slope': array(['New York'], dtype=object),
 'West Village': array(['New York', 'San Francisco', nan], dtype=object),
 'East Harlem': array(['New York', nan], dtype=object),
 'Fort Greene': array(['New York'], dtype=object),
 'Chelsea': array(['New York', nan], dtype=object),
 'Crown Heights': array(['New York', 'Paris', nan], dtype=object),
 'Park Slope': array(['New York', nan], dtype=object),
```

Fill the market using the neighbourhood_cleansed column

In [35]:
```
def impute_market(row):
    if type(row["market"]) != type("a"):
        nb_cleansed = row["neighbourhood_cleansed"]
        val = map_nbc_market[nb_cleansed][0]
        return val
    else:
        return row
    return row

listings["market"] = listings[["market", "neighbourhood_cleansed"]].apply(im
```

host_location and host_neighbourhood

In [36]:
```
listings["host_location"].fillna("XX", inplace=True)
```

In [37]:
```
listings["host_neighbourhood"].fillna("XX", inplace=True)
```

neighbourhood

It might be safer to set the missing values in this column with the same ones from "neighbourhood_cleansed"

In [38]:
```
listings["neighbourhood"].fillna(listings["neighbourhood_cleansed"], inplace
```

Zipcode

In [39]:
```
len(listings["zipcode"].unique())
```
Out[39]: 378

In [40]:
```
listings["zipcode"].fillna("XX", inplace=True)
```

First review and last review

Date columns. Why are there missing values?

In [10]:
```
review_date_cols = ["first_review", "last_review"]
nan_df = analyse_nans(listings)

# Verify that these columns have missing values in the exact same rows.
print("Number of rows between pairs of columns that have NaNs only in one of
for i in range(1, len(review_date_cols)):
    print(review_date_cols[i], "\t", review_date_cols[i-1], end=" -->\t ")
    idx1 = nan_df[review_date_cols[i-1]].iloc[2]
    idx2 = nan_df[review_date_cols[i]].iloc[2]
    l = find_unique_elems([idx1, idx2])
    print(len(l), "rows, \ti.e.", round(100*len(l)/listings.shape[0], 2), "%
```
```
Number of rows between pairs of columns that have NaNs only in one of them -
last_review      first_review -->        0 rows,        i.e. 0.0 % of total lis
tings rows
```

In [11]:
```
listings[(~listings["last_review"].isna())&(listings["first_review"].isna())
        (~listings["first_review"].isna())&(listings["last_review"].isna())]
```
Out[11]:

	listing_url	scrape_id	last_scraped	name	summary	space	description	experiences_offered	ı
id									

In [12]:
```
listings['last_review'] = pd.to_datetime(listings['last_review'])
listings['lreview_year'] = listings['last_review'].dt.year
listings['lreview_month'] = listings['last_review'].dt.month
listings['lreview_day'] = listings['last_review'].dt.day

listings['first_review'] = pd.to_datetime(listings['first_review'])
listings['freview_year'] = listings['first_review'].dt.year
listings['freview_month'] = listings['first_review'].dt.month
listings['freview_day'] = listings['first_review'].dt.day
```

In [13]:
```
cols = ["last_review", 'lreview_year', 'lreview_month', 'lreview_day']
listings[cols].describe()
```

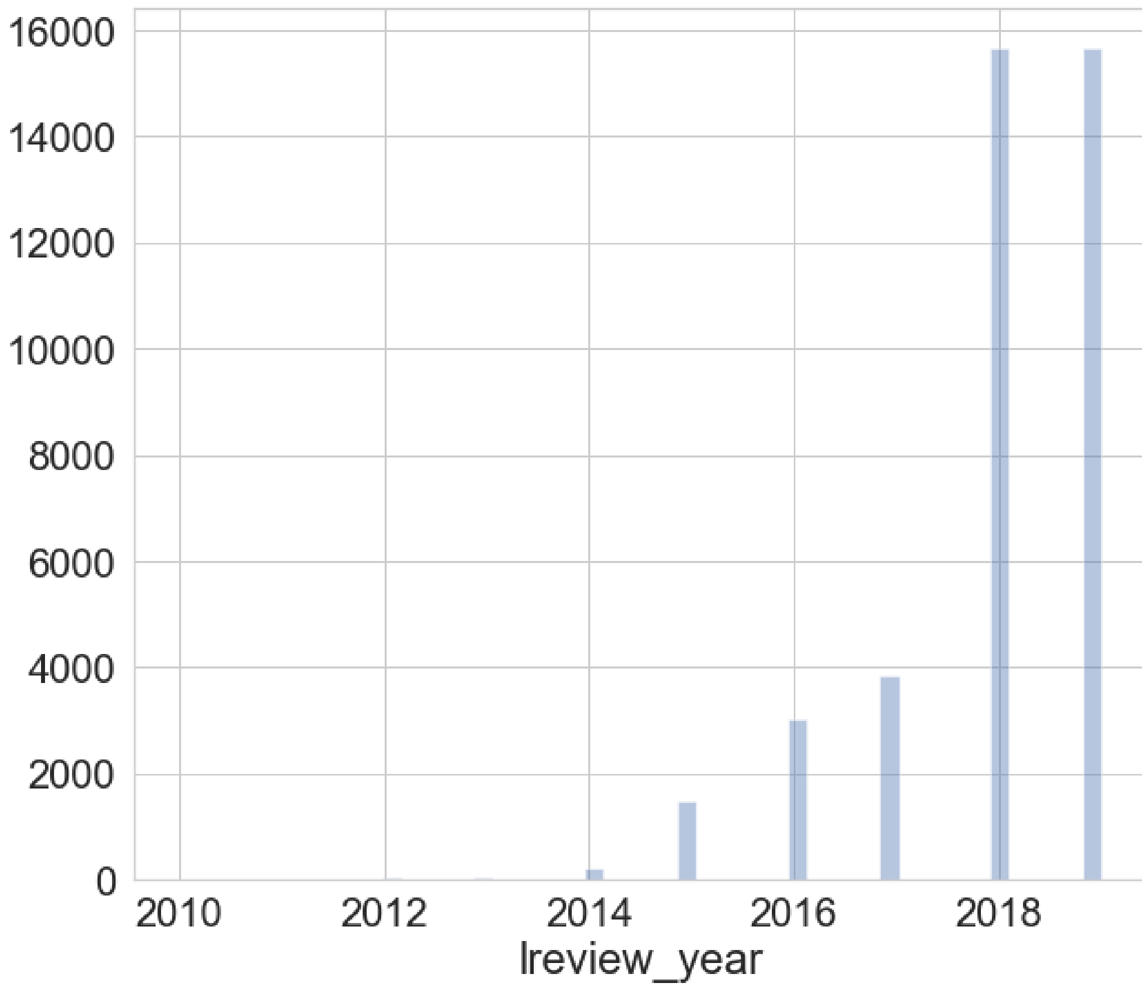
Out[13]:

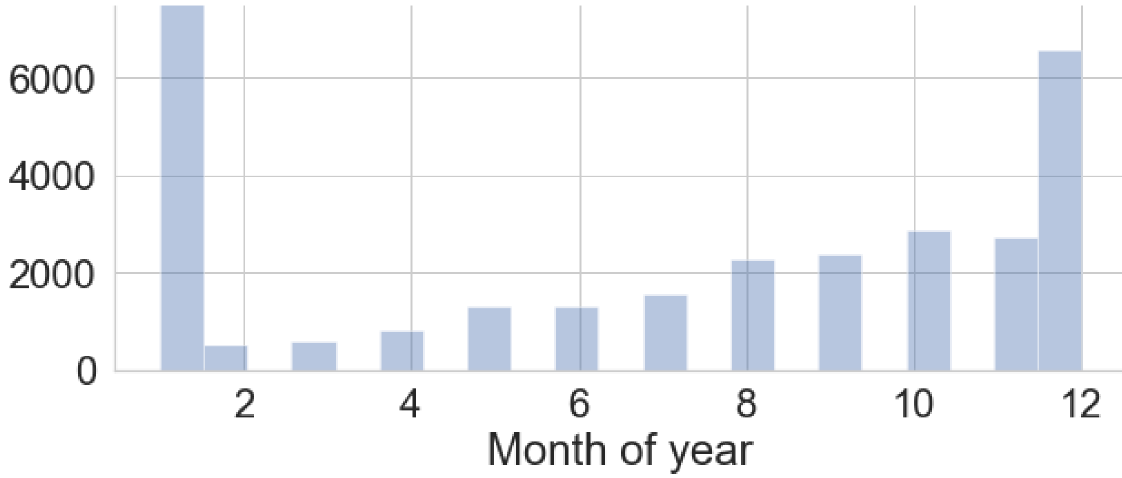
	lreview_year	lreview_month	lreview_day
count	39931.000000	39931.000000	39931.000000
mean	2017.998247	5.644687	14.929178
std	1.133024	4.536616	10.082954
min	2010.000000	1.000000	1.000000
25%	2018.000000	1.000000	5.000000
50%	2018.000000	5.000000	15.000000
75%	2019.000000	10.000000	24.000000
max	2019.000000	12.000000	31.000000

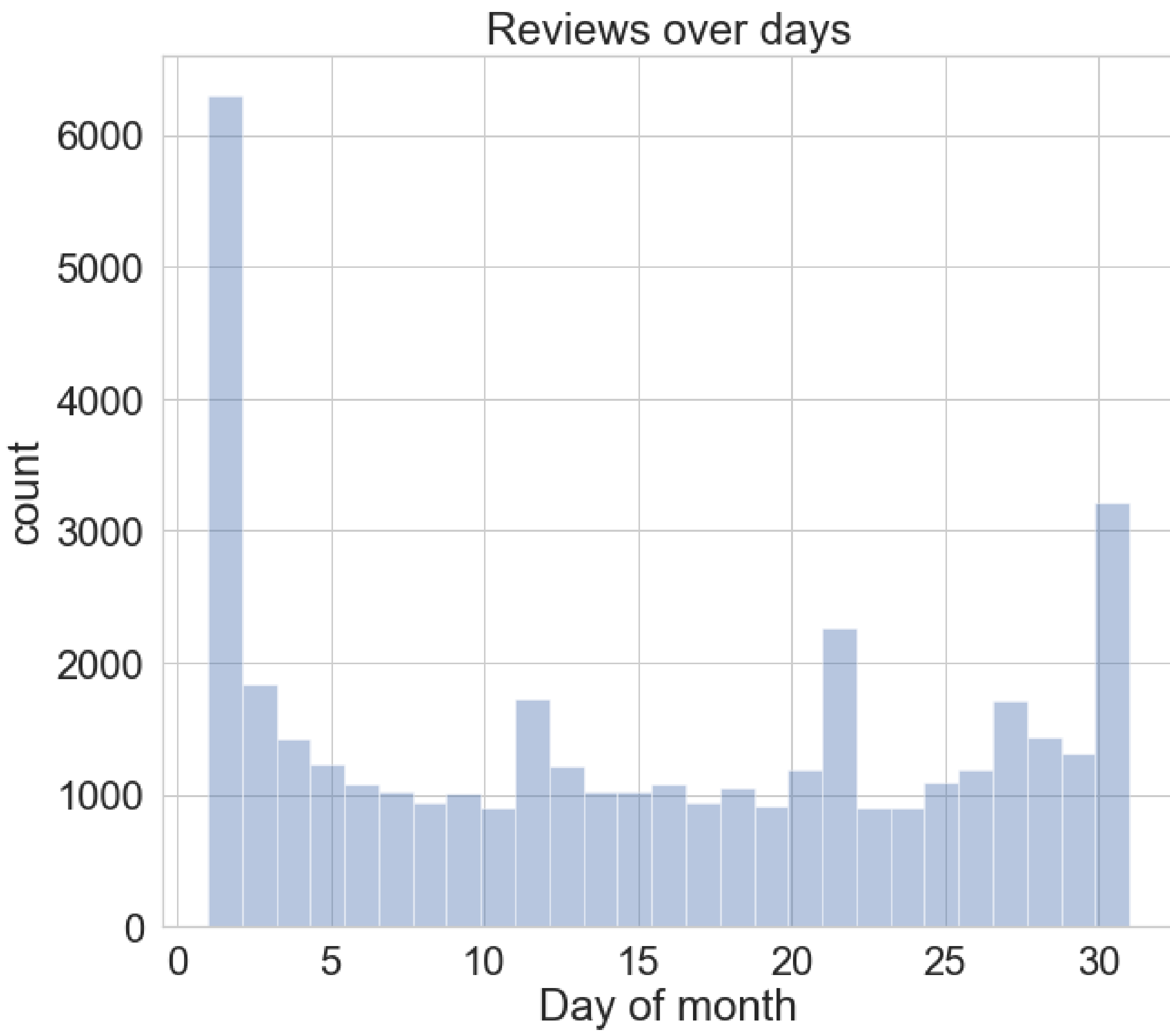
In [14]:
```
sns.set(font_scale=2, style="whitegrid")
```

In [16]:
```python
f, ax = plt.subplots(3, 1, figsize=(10, 30))
g = sns.distplot(a=listings[~listings["lreview_year"].isna()]["lreview_year"
g = sns.distplot(a=listings[~listings["lreview_month"].isna()]["lreview_mont
g.set(title="Reviews over months", xlabel="Month of year", ylabel="count")
g = sns.distplot(a=listings[~listings["lreview_day"].isna()]["lreview_day"],
g.set(title="Reviews over days", xlabel="Day of month", ylabel="count")
```

Out[16]:
```
[Text(0, 0.5, 'count'),
 Text(0.5, 0, 'Day of month'),
 Text(0.5, 1.0, 'Reviews over days')]
```







If we use median, we can estimate that the missing year values is (most probably) 2018, the missing month values is (most probably) 10/11 and for date would be around 16.

Or we can create a separate category called X for each column, to estimate that there has been no review written for those dates yet.Upon checking online using the URLs provided, we can verify that no reviews are provided for those listings yet.

The latter method is used for imputation here.

In [47]:
```
listings["lreview_year"].apply(type)
```

Out[47]:
```
0       <class 'float'>
1       <class 'float'>
2       <class 'float'>
3       <class 'float'>
4       <class 'float'>
5       <class 'float'>
6       <class 'float'>
7       <class 'float'>
8       <class 'float'>
9       <class 'float'>
10      <class 'float'>
11      <class 'float'>
12      <class 'float'>
13      <class 'float'>
14      <class 'float'>
15      <class 'float'>
16      <class 'float'>
17      <class 'float'>
18      <class 'float'>
```

In [49]:
```
listings["first_review"].fillna("XX", inplace=True)
listings["last_review"].fillna("XX", inplace=True)
listings['lreview_year'].fillna(0, inplace=True)
listings['lreview_month'].fillna(0, inplace=True)
listings['lreview_day'].fillna(0, inplace=True)
listings['freview_year'].fillna(0, inplace=True)
listings['freview_month'].fillna(0, inplace=True)
listings['freview_day'].fillna(0, inplace=True)
```

In [54]:
```
listings['ndays_between_f_l_reviews'] = abs(listings['lreview_day'] - listin
```

In [ ]:
```
listings
```

In [55]:
```
listings['ndays_between_f_l_reviews'].head()
```

Out[55]:
```
0     0.0
1    15.0
2    19.0
3     3.0
4     0.0
Name: ndays_between_f_l_reviews, dtype: float64
```

In [51]:
```
1 plot_nans(listings)
```