In [ ]:

```
```

Data fields train.tsv, test.tsv The files consist of a list of product listings. These files are tab-delimited.

train_id or test_id - the id of the listing

name - the title of the listing. Note that we have cleaned the data to remove text that look like prices (e.g. $20) to avoid leakage. These removed prices are represented as [rm]

item_condition_id - the condition of the items provided by the seller

category_name - category of the listing

brand_name

price - the price that the item was sold for. This is the target variable that you will predict. The unit is USD. This column doesn't exist in test.tsv since that is what you will predict.

shipping - 1 if shipping fee is paid by seller and 0 by buyer

item_description - the full description of the item. Note that we have cleaned the data to remove text that look like prices (e.g. $20) to avoid leakage. These removed prices are represented as [rm]

Refrences:
https://www.kaggle.com/konohayui/mercari-price-suggestion-eda
https://www.kaggle.com/thykhuely/mercari-interactive-eda-topic-modelling

Other References:
https://medium.com/@karthiktsaliki/automated-way-for-predicting-price-of-products-414f70df2b8a
https://medium.com/unstructured/how-i-lost-a-silver-medal-in-kagglesmercari-price-suggestion-challenge-using-cnns-and-tensorflow-4013660fcded

In [ ]:

```
```

In [ ]:

```
import pandas as pd
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import warnings
import warnings
#warnings.filterwarnings("ignore")
import shutil
import os
import pickle
from sklearn.manifold import TSNE
import multiprocessing
import codecs
import random as r
```

In [ ]:

```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.calibration import CalibratedClassifierCV
from sklearn.metrics import log_loss
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

Reading the data

In [ ]:

```
train=pd.read_csv('train.tsv',sep='\t')
test=pd.read_csv('test.tsv',sep='\t')
```

In [ ]:

```
train.head(5)
```

Out[ ]:

	train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	NaN	10.0	1	No description yet
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	0	This keyboard is in great condition and works ...
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	1	Adorable top with a hint of lace and a key hol...
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	NaN	35.0	1	New with tags. Leather horses. Retail for [rm]...
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	NaN	44.0	0	Complete with certificate of authenticity

In [ ]:

```
test.head()
```

Out[ ]:

	test_id	name	item_condition_id	category_name	brand_name	shipping	item_description
0	0	Breast cancer "I fight like a girl" ring	1	Women/Jewelry/Rings	NaN	1	Size 7
1	1	25 pcs NEW 7.5"x12" Kraft Bubble Mailers	1	Other/Office supplies/Shipping Supplies	NaN	1	25 pcs NEW 7.5"x12" Kraft Bubble Mailers Lined...
2	2	Coach bag	1	Vintage & Collectibles/Bags and Purses/Handbag	Coach	1	Brand new coach bag. Bought for [rm] at a Coac...
3	3	Floral Kimono	2	Women/Sweaters/Cardigan	NaN	0	-floral kimono -never worn -lightweight and pe...
4	4	Life after Death	3	Other/Books/Religion & Spirituality	NaN	1	Rediscovering life after the loss of a loved o...

In [ ]:

```

```

Checking the type of features

In [ ]:

```
train.dtypes
```

Out[ ]:

```
train_id               int64
name                  object
item_condition_id       int64
category_name         object
brand_name            object
price                float64
shipping               int64
item_description      object
dtype: object
```

In [ ]:

```
test.dtypes
```

Out[ ]:

```
test_id                int64
name                  object
item_condition_id       int64
category_name         object
brand_name            object
shipping               int64
item_description      object
dtype: object
```

In [ ]:

```

```

The Shape of the data

In [ ]:

```
print('Number of data points : ', train.shape[0])
print('Number of features : ', train.shape[1])
print('The features are : ',train.columns.values)
```

```
Number of data points :  1482535
Number of features :  8
The features are :  ['train_id' 'name' 'item_condition_id' 'category_name' 'brand_name'
 'price' 'shipping' 'item_description']
```

In [ ]:

```
print('Number of data points : ', test.shape[0])
print('Number of features : ', test.shape[1])
print('The features are : ',test.columns.values)
# There is no price column in the test dataset as we have to predict it
```

```
Number of data points :  693359
Number of features :  7
The features are :  ['test_id' 'name' 'item_condition_id' 'category_name' 'brand_name'
 'shipping' 'item_description']
```

In [ ]:

```

```

In [ ]:

```

```

The columns in the given data containing null values

In Train Data

In [ ]:

```
train[train.isnull().any(axis=1)]
```

Out[ ]:

	train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	NaN	10.0	1	No description yet
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	NaN	35.0	1	New with tags. Leather horses. Retail for [rm]...
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	NaN	44.0	0	Complete with certificate of authenticity
5	5	Bundled items requested for Ruie	3	Women/Other/Other	NaN	59.0	0	Banana republic bottoms, Candies skirt with ma...
9	9	Porcelain clown doll checker pants VTG	3	Vintage & Collectibles/Collectibles/Doll	NaN	8.0	0	I realized his pants are on backwards after th...
...	...	...	...	...	...	...	...	...
1482526	1482526	Harry Potter Shirt! Women M/ Girl XL	2	Women/Tops & Blouses/T-Shirts	NaN	12.0	0	Great Harry Potter Shirt! "Hogwarts, School of...
1482527	1482527	Blk/white ribbed mock neck bodysuit M	1	Women/Tops & Blouses/Blouse	NaN	10.0	1	Brand new black and white ribbed mock neck bod...
1482532	1482532	21 day fix containers and eating plan	2	Sports & Outdoors/Exercise/Fitness accessories	NaN	12.0	0	Used once or twice, still in great shape.
1482533	1482533	World markets lanterns	3	Home/Home Décor/Home Décor Accents	NaN	45.0	1	There is 2 of each one that you see! So 2 red ...
1482534	1482534	Brand new lux de ville wallet	1	Women/Women's Accessories/Wallets	NaN	22.0	0	New with tag, red with sparkle. Firm price, no...

635553 rows × 8 columns

From the above result we can say that out of 148253 rows, there are 635553 rows containing null values.

In [ ]:

```

```

In Test Data

In [ ]:

```
test[test.isnull().any(axis=1)]
```

Out[ ]:

	test_id	name	item_condition_id	category_name	brand_name	shipping	item_description
0	0	Breast cancer "I fight like a girl" ring	1	Women/Jewelry/Rings	NaN	1	Size 7
1	1	25 pcs NEW 7.5"x12" Kraft Bubble Mailers	1	Other/Office supplies/Shipping Supplies	NaN	1	25 pcs NEW 7.5"x12" Kraft Bubble Mailers Lined...
3	3	Floral Kimono	2	Women/Sweaters/Cardigan	NaN	0	-floral kimono - never worn - lightweight and pe...
4	4	Life after Death	3	Other/Books/Religion & Spirituality	NaN	1	Rediscovering life after the loss of a loved o...
5	5	iPhone 6 Plus or 6s Plus Vodka pink case	1	Electronics/Cell Phones & Accessories/Cases, C...	NaN	1	One Absolut Vodka in Pink for iPhone 6 Plus an...
...	...	...	...	...	...	...	...
693350	693350	NEW FIDGET HAND SPINNER DESK TOY CUBE	1	Kids/Toys/Games	NaN	1	[rm] free shipping too New and highly addictive
693354	693354	Quartz crystal on Flint stone	1	Home/Home Décor/Home Décor Accents	NaN	0	Flint/Quartz cluster. Self mined measures 3x2...
693356	693356	Galaxy S8 hard shell case	1	Electronics/Cell Phones & Accessories/Cases, C...	NaN	1	New. Free shipping Basstop case
693357	693357	Hi low floral kimono	2	Women/Swimwear/Cover-Ups	NaN	0	Floral kimono. Tropical print. Open front. Hi ...
693358	693358	FREESHIP 2 Floral Scrub Tops, medium.	2	Women/Tops & Blouses/T-Shirts	NaN	1	2 Floral scrub tops. Worn less than 5 times ea...

296928 rows × 7 columns

From the above result we can say that out of 693359 rows, there are 296928 rows containing null values.

The Count of Null values in those features

In [ ]:

```
train.isnull().sum()
```

Out[ ]:

```
train_id                0
name                    0
item_condition_id        0
category_name         6327
brand_name          632682
price                    0
shipping                 0
```

```
item_description           4
dtype: int64
```

In [ ]:

```
test.isnull().sum()
```

Out[ ]:

```
test_id                    0
name                       0
item_condition_id          0
category_name           3058
brand_name            295525
shipping                   0
item_description           0
dtype: int64
```

Analysis of Individual Features

First let us check for the Price(Target) Variable

In [ ]:

```
train['price'].describe()
```

Out[ ]:

```
count    1.482535e+06
mean     2.673752e+01
std      3.858607e+01
min      0.000000e+00
25%      1.000000e+01
50%      1.700000e+01
75%      2.900000e+01
max      2.009000e+03
Name: price, dtype: float64
```

Univariate Distribution of "Price" Variable

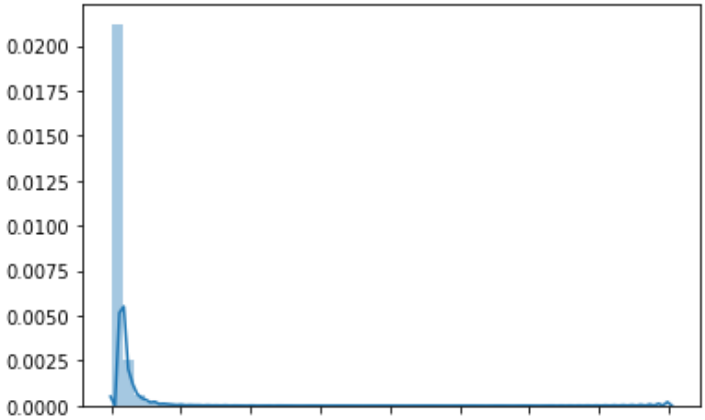
In [ ]:

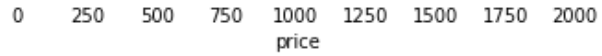
```
sns.distplot(train['price'])
```

```
C:\Users\yacca\Anaconda\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c6935b5f48>
```

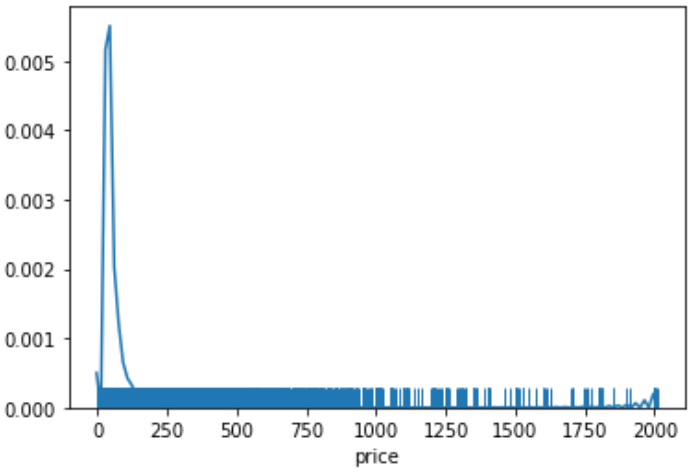




Kernel Density Estimate of "Price"

In [ ]:

```
sns.distplot(train['price'], hist=False, rug=True);
```



From the first plot, we can see that there are more products whose prices are ranging from 0 to 230 dollars and few prodcuts in range of 1800-2000. But from the second plot, we can say that there are products having prices from 250 to 2000 dollars.

In [ ]:

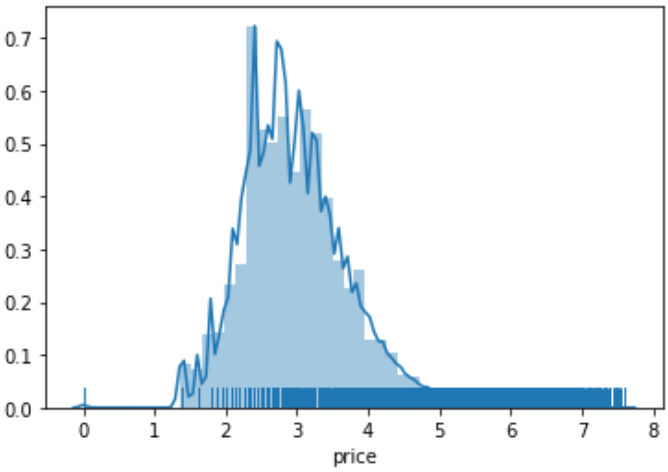
```

```

Just added +1 to reduce the log error

In [ ]:

```
sns.distplot(np.log(train['price']+1), hist=True, rug=True);
```



In [ ]:

```
print("There are", train[train["price"] == 0].price.size, "items with price 0.")
```

```
There are 874 items with price 0.
```

In [ ]:

```

```

In [ ]:

```

```

Categories Feature

We have seen from the above method that there are 6327 with null values in the category_name feature

Series containing counts of unique values

In [ ]:

```
train['category_name'].unique()
```

Out[ ]:

```
array(['Men/Tops/T-shirts',
       'Electronics/Computers & Tablets/Components & Parts',
       'Women/Tops & Blouses/Blouse', ..., 'Handmade/Jewelry/Clothing',
       'Vintage & Collectibles/Supplies/Ephemera',
       'Handmade/Pets/Blanket'], dtype=object)
```

In [ ]:

```
train['category_name'].value_counts().shape
```

Out[ ]:

```
(1287,)
```

We can see that there are 1287 Unique categories available

In [ ]:

```
train['category_name'].value_counts()[:10]
```

Out[ ]:

```
Women/Athletic Apparel/Pants, Tights, Leggings                 60177
Women/Tops & Blouses/T-Shirts                                  46380
Beauty/Makeup/Face                                             34335
Beauty/Makeup/Lips                                             29910
Electronics/Video Games & Consoles/Games                       26557
Beauty/Makeup/Eyes                                             25215
Electronics/Cell Phones & Accessories/Cases, Covers & Skins    24676
Women/Underwear/Bras                                           21274
Women/Tops & Blouses/Blouse                                    20284
Women/Tops & Blouses/Tank, Cami                                20284
Name: category_name, dtype: int64
```

6327 fields in the 'category_name' has null values.

In [ ]:

```
train.shape
```

Out[ ]:

```
(1482535, 8)
```

In [ ]:

```
train.isnull().sum()
```

Out[ ]:

```
train_id                0
name                    0
item_condition_id        0
category_name         6327
brand_name          632682
price                   0
```

```
shipping                  0
item_description          4
dtype: int64
```

In [ ]:

```
```

Filling missing data

In [ ]:

```
def fill_missing_data(data):
    data.category_name.fillna(value = "Other/Other/Other", inplace = True)
    data.brand_name.fillna(value = "Unknown", inplace = True)
    data.item_description.fillna(value = "No description yet", inplace = True)
    return data

train = fill_missing_data(train)
```

In [ ]:

```
train[11000:11010]
```

Out[ ]:

	train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description
11000	11000	New Lancome Rénergie Lift & Visionnaire	1	Beauty/Skin Care/Face	Lancome	115.0	1	Bundle Lancome Rénergie Lift & Visionnaire Cx ...
11001	11001	Puma bodysuit 5pk	1	Kids/Girls 0-24 Mos/One-Pieces	Unknown	14.0	0	Brand new Puma 5 pack Size 3-6M
11002	11002	Pacifier lot advent soothie nuk gerber	3	Kids/Feeding/Pacifiers & Accessories	Avent	6.0	1	7 used pacifiers good condition 1 soothie has ...
11003	11003	Michael kors	3	Women/Women's Handbags/Shoulder Bag	Michael Kors	45.0	1	Michael kors purse some stains in the inside n...
11004	11004	LuLaRoe Perfect T GUC L	3	Women/Tops & Blouses/T-Shirts	Unknown	14.0	1	Good used condition, worn 2-3 times but washed...
11005	11005	Red cheetah purse	2	Women/Women's Handbags/Shoulder Bag	Neiman Marcus	8.0	1	No description yet
11006	11006	Pandora Delicate Sentiments Bow Ring	2	Women/Jewelry/Rings	PANDORA	35.0	1	PANDORA Delicate Sentiments Ring with Clear Cu...
11007	11007	Drop Hoop Earrings	2	Women/Jewelry/Earrings	Unknown	5.0	1	purchased from Dillards. Like new. Comes in bo...
11008	11008	Pink 7 plus Card Holder Phone Case	1	Women/Women's Accessories/Other	PINK	19.0	0	No description yet
11009	11009	Head and Sholders	1	Beauty/Hair Care/Shampoo Plus Conditioner	Unknown	17.0	0	4 bottles Read my profile as I will be unavail...

In [ ]:

```
train.isnull().sum()
```

```
train.isnull().sum()
```

Out[ ]:

```
train_id             0
name                 0
item_condition_id     0
category_name        0
brand_name           0
price                0
shipping             0
item_description     0
dtype: int64
```

There are no null values now in the dataset so that we can move on further

In [ ]:

```

```

Splitting into Sub-Categories

In [ ]:

```
def split_cat(text):
    try: return text.split("/")
    except: return ("No Label", "No Label", "No Label")
```

In [ ]:

```
train['general_cat'], train['subcat_1'], train['subcat_2'] = \
zip(*train['category_name'].apply(lambda x: split_cat(x)))
train.head()
```

Out[ ]:

	train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description	general_
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	Unknown	10.0	1	No description yet	I
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	0	This keyboard is in great condition and works ...	Electro
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	1	Adorable top with a hint of lace and a key hol...	Wo
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	Unknown	35.0	1	New with tags. Leather horses. Retail for [rm]...	H
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	Unknown	44.0	0	Complete with certificate of authenticity	Wo

In [ ]:

```
train.isnull().sum()
```

Out[ ]:

```
train_id             0
name                 0
item_condition_id     0
category_name        0
brand_name           0
```

```
brand_name          0
price               0
shipping            0
item_description    0
general_cat         0
subcat_1            0
subcat_2            0
dtype: int64
```

There are no null values in the dataset

In [ ]:

```
```

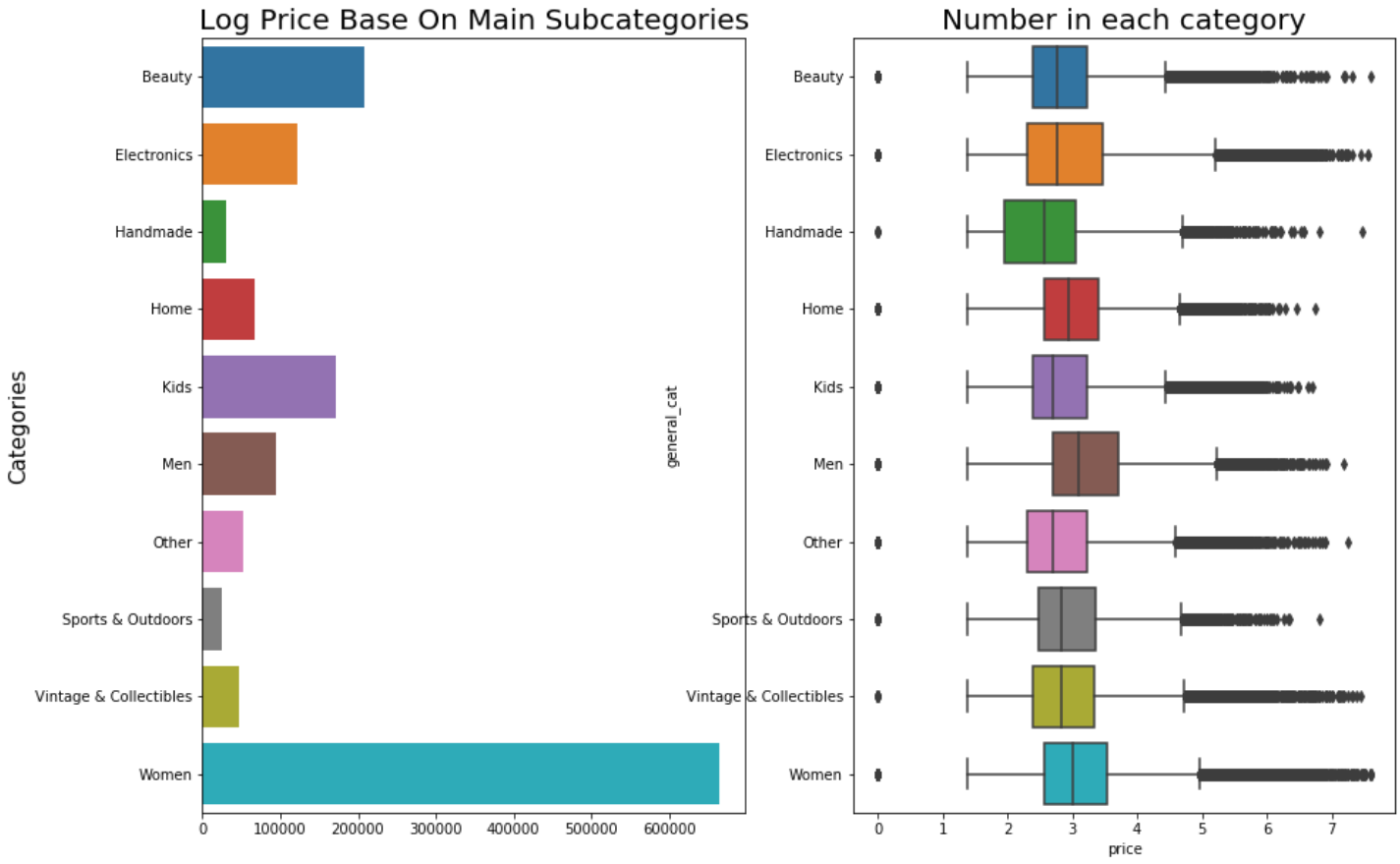
Price with respect to general category

In [ ]:

```
genreal_category=train['general_cat']
log_price=np.log(train['price']+1)
```

In [ ]:

```
order = sorted(genreal_category.unique())
fig, ax = plt.subplots(1, 2, figsize = (15, 10))
sns.boxplot(x = log_price, y = genreal_category, orient = "h", order = order, ax = ax[1]
)
ax[0].set_title("Log Price Base On Main Subcategories", fontsize = 20)
ax[0].set_ylabel("Categories", fontsize = 15)
sns.barplot(genreal_category.value_counts().values, genreal_category.value_counts().inde
x, order = order, ax = ax[0])
ax[1].set_title("Number in each category", fontsize = 20)
plt.show()
```



In [ ]:

```
order
```

Out[ ]:

```
['Beauty',
 'Electronics',
 'Handmade',
 'Home',
 'Kids',
 'Men',
 'Other',
 'Sports & Outdoors',
 'Vintage & Collectibles',
 'Women']
```

In [ ]:

```
```

Price with respect to sub- category

In [ ]:

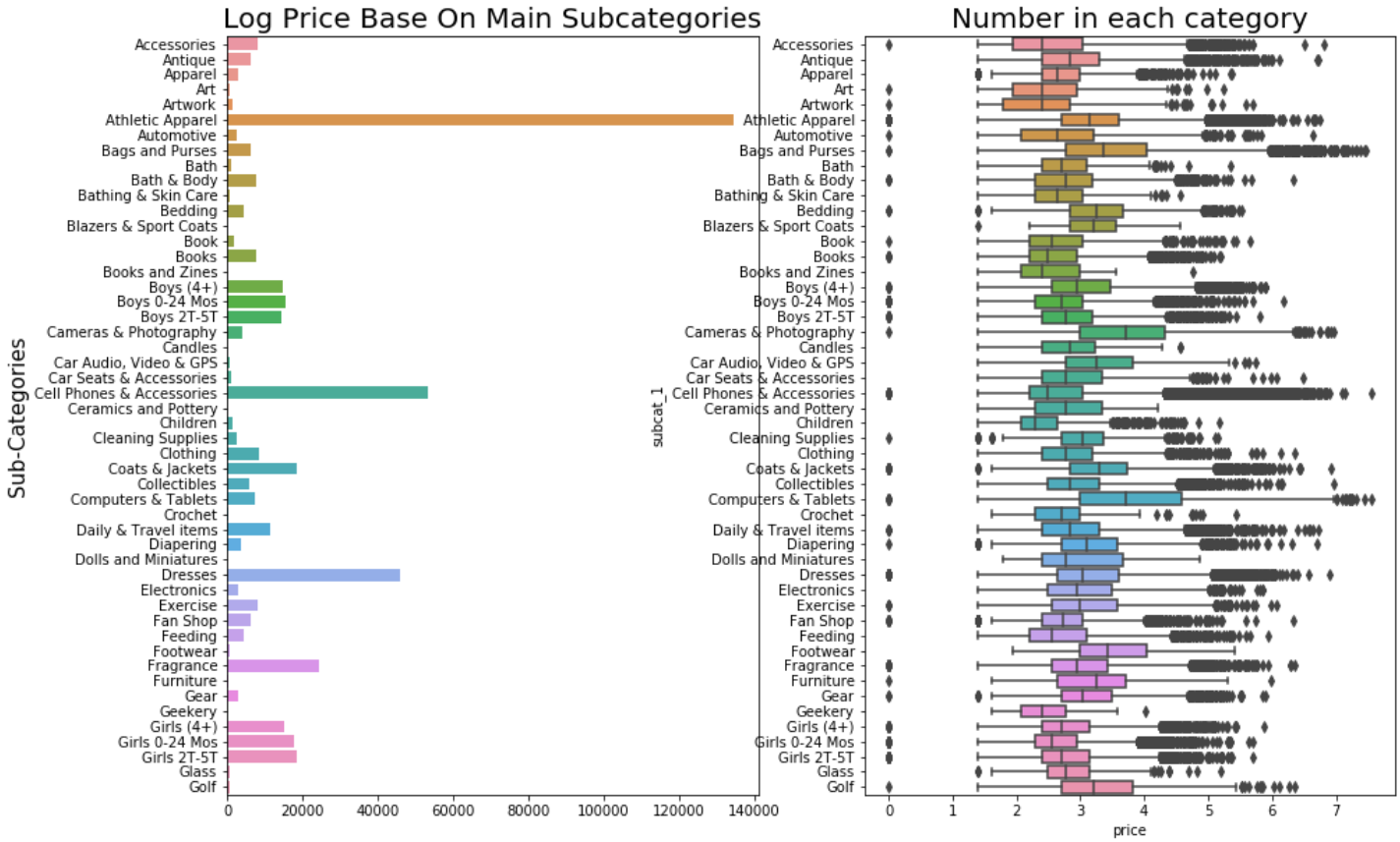
```
```

In [ ]:

```
subcat1=train['subcat_1']
```

In [ ]:

```
order1 = sorted(subcat1.unique())
fig, ax = plt.subplots(1, 2, figsize = (15, 10))
sns.boxplot(x = log_price, y = subcat1, orient = "h", order = order1[0:50], ax = ax[1])
ax[0].set_title("Log Price Base On Main Subcategories", fontsize = 20)
ax[0].set_ylabel("Sub-Categories", fontsize = 15)
sns.barplot(subcat1.value_counts().values, subcat1.value_counts().index, order = order1[
0:50], ax = ax[0])
ax[1].set_title("Number in each category", fontsize = 20)
plt.show()
```



In [ ]:

```
```

In [ ]:

```
train.columns
```

Out[ ]:

```
Index(['train_id', 'name', 'item_condition_id', 'category_name', 'brand_name',
       'price', 'shipping', 'item_description', 'general_cat', 'subcat_1',
       'subcat_2'],
      dtype='object')
```

In [ ]:

```

```

Variation of price with respect to Brand

In [ ]:

```

```

In [ ]:

```
brands = train["brand_name"].value_counts()
brands[:13]
```

Out[ ]:

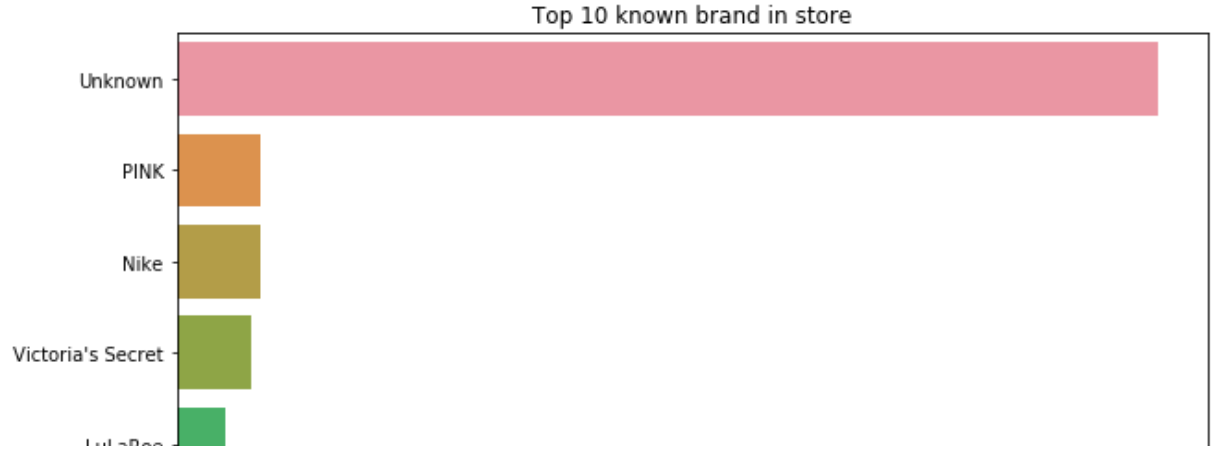
```
Unknown              632682
PINK                  54088
Nike                  54043
Victoria's Secret     48036
LuLaRoe               31024
Apple                 17322
FOREVER 21            15186
Nintendo              15007
Lululemon             14558
Michael Kors          13928
American Eagle        13254
Rae Dunn              12305
Sephora               12172
Name: brand_name, dtype: int64
```

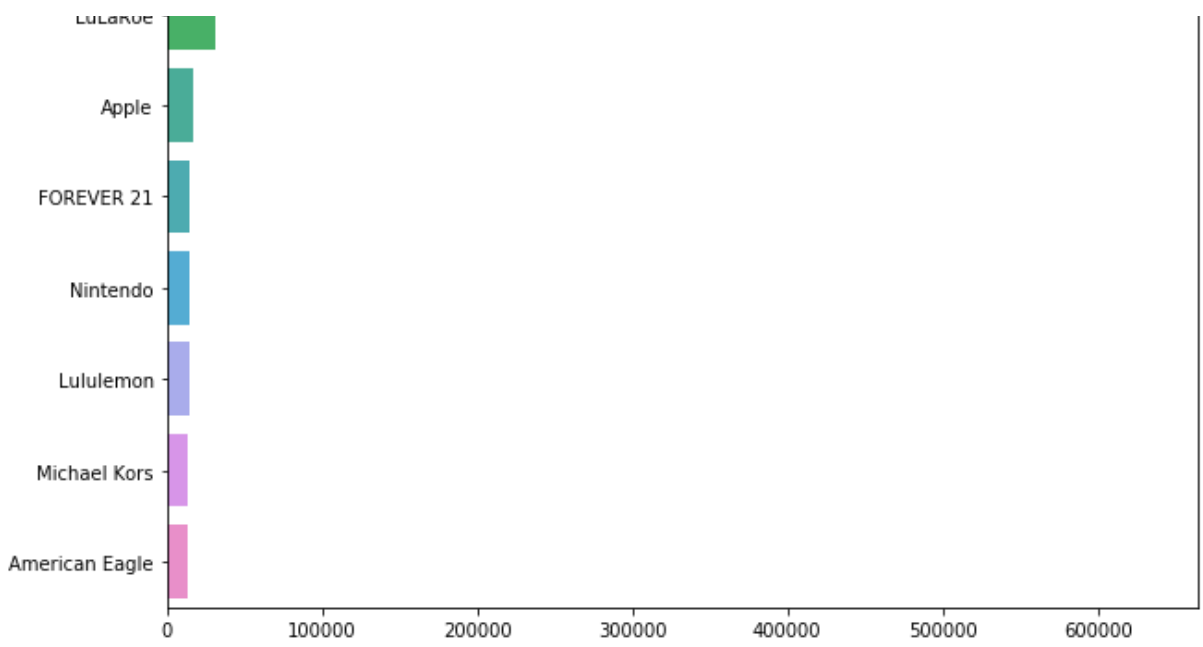
In [ ]:

```
brand = brands[0:12]
log_price=np.log(train['price']+1)
```

In [ ]:

```
plt.figure(figsize = (10, 10))
sns.barplot(brand[0:11].values, brand[0:11].index)
plt.title("Top 10 known brand in store")
plt.show()
```





In [ ]:

```
```

In [ ]:

```
train[train["brand_name"] == "Unknown"].price.describe()
```

Out[ ]:

```
count    632682.000000
mean         21.133453
std          27.361260
min           0.000000
25%           9.000000
50%          14.000000
75%          24.000000
max        2000.000000
Name: price, dtype: float64
```

In [ ]:

```
```

Item description

In [ ]:

```
import tokenize
import nltk
from nltk import tokenize
```

In [ ]:

```
train['item_description'].head()
```

Out[ ]:

```
0                                   No description yet
1    This keyboard is in great condition and works ...
2    Adorable top with a hint of lace and a key hol...
3    New with tags. Leather horses. Retail for [rm]...
4              Complete with certificate of authenticity
Name: item_description, dtype: object
```

In [ ]:

```
```

Preprocessing the text

In [ ]:

```
import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
stop = set(stopwords.words('english')) #set of stopwords
sno = nltk.stem.SnowballStemmer('english') #initialising the snowball stemmer
def cleanhtml(sentence): #function to clean the word of any html-tags
 cleanr = re.compile('<.*?>')
 cleantext = re.sub(cleanr, ' ', sentence)
 return cleantext
def cleanpunc(sentence): #function to clean the word of any punctuationor special charact
ers
 cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
 cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
 return cleaned
print(stop)
print('************************************')
```

```
{'below', 'once', "wouldn't", 'these', "hadn't", 'd', 'its', 'such', 'then', 'further', '
why', 'doing', 'himself', 'the', 'but', 'had', "aren't", "should've", 've', 'your', 'and'
, "shouldn't", 'during', 'hers', 'off', 'is', "doesn't", 'or', 'just', 'weren', 'been', '
have', 'should', 'needn', "shan't", "weren't", 'herself', 'wouldn', "you'll", 'our', 'you
rself', 'against', 'in', "it's", 'more', 'between', 'were', 'how', 'than', 'won', 'not',
'will', 'very', 'i', 'does', "you'd", 'of', 'here', 'who', 'nor', 'yourselves', "you've",
'any', 'into', 'hadn', "needn't", 'so', "mightn't", 'over', 'me', 'has', 'ours', "didn't"
, 'at', 'same', 'to', 'their', 'having', 'on', 'before', 'only', 'that', 'whom', 'can', '
his', 'her', 'with', 'out', 're', 'he', 'itself', 'because', 'both', 'my', "isn't", 'bein
g', 'until', 'what', 'you', 'which', 'ma', 'ourselves', 'ain', 'isn', 'an', 'about', 'm',
'those', 'mustn', 'down', 'didn', 'did', 'each', 'other', 'aren', 'll', 'it', 'yours', 'a
bove', 'a', 'do', 'by', 'shan', 'from', 'up', 'after', "wasn't", 's', 'o', 'there', "she'
s", 'where', 'for', 'we', 'theirs', 'own', 'them', 'don', 'was', 'be', 'hasn', 'they', "h
aven't", 'mightn', 'this', 'while', 'him', 'when', "mustn't", 'haven', 'wasn', 'few', 't'
, 'doesn', 'under', 'all', 'couldn', 'as', 'she', 'no', "you're", 'y', 'most', 'shouldn',
"don't", 'too', "couldn't", "won't", 'if', 'am', 'are', "that'll", 'again', 'now', 'some'
, 'themselves', 'through', "hasn't", 'myself'}
************************************
```

In [ ]:

```
i=0
str1=' '
final_string=[]
all_positive_words=[] # store words from +ve reviews here
all_negative_words=[] # store words from -ve reviews here.
s=''
for sent in train['item_description'].values:
    filtered_sentence=[]
    #print(sent);
    sent=cleanhtml(sent) # remove HTML tags
    for w in sent.split():
        for cleaned_words in cleanpunc(w).split():
            if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                if(cleaned_words.lower() not in stop):
                    s=(sno.stem(cleaned_words.lower())).encode('utf8')
                    filtered_sentence.append(s)
                    if (train['item_description'].values)[i] == 'positive':
                        all_positive_words.append(s) #list of all words used to describe
positive reviews
                    if(train['item_description'].values)[i] == 'negative':
                        all_negative_words.append(s) #list of all words used to describe
negative reviews reviews
                else:
                    continue
```

```
            else:
                continue
    #print(filtered_sentence)
    str1 = b" ".join(filtered_sentence) #final string of cleaned words
    #print("**************************************************************")

    final_string.append(str1)
    i+=1
```

In [ ]:

```
train['desc']=final_string
```

In [ ]:

```
train['desc'].head()
```

Out[ ]:

```
0                                    b'descript yet'
1    b'keyboard great condit work like came box por...
2    b'ador top hint lace key hole back pale pink a...
3    b'new tag leather hors retail stand foot high ...
4                           b'complet certif authent'
Name: desc, dtype: object
```

In [ ]:

```

```

In [ ]:

```
'''i=0
str1=' '
final_string=[]
all_positive_words=[] # store words from +ve reviews here
all_negative_words=[] # store words from -ve reviews here.
s=''
for sent in test['item_description'].values:
    filtered_sentence=[]
    #print(sent);
    sent=cleanhtml(sent) # remove HTML tags
    for w in sent.split():
        for cleaned_words in cleanpunc(w).split():
            if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                if(cleaned_words.lower() not in stop):
                    s=(sno.stem(cleaned_words.lower())).encode('utf8')
                    filtered_sentence.append(s)
                    if (test['item_description'].values)[i] == 'positive':
                        all_positive_words.append(s) #list of all words used to describe
positive reviews
                    if(test['item_description'].values)[i] == 'negative':
                        all_negative_words.append(s) #list of all words used to describe
negative reviews reviews
                else:
                    continue
            else:
                continue
    #print(filtered_sentence)
    str1 = b" ".join(filtered_sentence) #final string of cleaned words
    #print("**************************************************************")

    final_string.append(str1)
    i+=1'''
```

Out[ ]:

```
'i=0\nstr1=\' \'\nfinal_string=[]\nall_positive_words=[] # store words from +ve reviews h
ere\nall_negative_words=[] # store words from -ve reviews here.\ns=\'\'\nfor sent in test
[\'item_description\'].values:\n    filtered_sentence=[]\n    #print(sent);\n    sent=cle
anhtml(sent) # remove HTMl tags\n    for w in sent.split():\n        for cleaned_words in
cleanpunc(w).split():\n            if((cleaned_words.isalpha()) & (len(cleaned_words)>2))
```

```
:    \n                if(cleaned_words.lower() not in stop):\n                    s=(sn
o.stem(cleaned_words.lower())).encode(\'utf8\')\n                    filtered_sentence.ap
pend(s)\n                    if (test[\'item_description\'].values)[i] == \'positive\': \
n                    all_positive_words.append(s) #list of all words used to describe
positive reviews\n                    if(test[\'item_description\'].values)[i] == \'negat
ive\':\n                        all_negative_words.append(s) #list of all words used to d
escribe negative reviews reviews\n                else:\n                    continue\n
else:\n                continue \n    #print(filtered_sentence)\n    str1 = b" ".join(fil
tered_sentence) #final string of cleaned words\n    #print("****************************
****************************************")\n    \n    final_string.append(str1)\n    i+
=1'
```

In [ ]:

Creating tokenize function

In [ ]:

```
from nltk.tokenize import sent_tokenize,word_tokenize
```

In [ ]:

```
stop = set(stopwords.words('english'))
def tokenize(text):
    """
    sent_tokenize(): segment text into sentences
    word_tokenize(): break sentences into words
    """
    try:
        regex = re.compile('[' +re.escape(string.punctuation) + '0-9\\r\\t\\n]')
        text = regex.sub(" ", text) # remove punctuation

        tokens_ = [word_tokenize(s) for s in sent_tokenize(text)]
        tokens = []
        for token_by_sent in tokens_:
            tokens += token_by_sent
        tokens = list(filter(lambda t: t.lower() not in stop, tokens))
        filtered_tokens = [w for w in tokens if re.search('[a-zA-Z]', w)]
        filtered_tokens = [w.lower() for w in filtered_tokens if len(w)>=3]

        return filtered_tokens

    except TypeError as e: print(text,e)
```

Just checking for the non-preprocessed values

In [ ]:

```
train['tokens'] = train['item_description'].map(tokenize)
test['tokens'] = test['item_description'].map(tokenize)
```

In [ ]:

```
for description, tokens in zip(train['item_description'].head(),
                               train['tokens'].head()):
    print('description:', description)
    print('tokens:', tokens)
```

```
description: No description yet
tokens: ['description', 'yet']
description: This keyboard is in great condition and works like it came out of the box. A
ll of the ports are tested and work perfectly. The lights are customizable via the Razer
Synapse app on your PC.
tokens: ['keyboard', 'great', 'condition', 'works', 'like', 'came', 'box', 'ports', 'test
ed', 'work', 'perfectly', 'lights', 'customizable', 'via', 'razer', 'synapse', 'app']
description: Adorable top with a hint of lace and a key hole in the back! The pale pink i
s a 1X, and I also have a 3X available in white!
```

```
s a 1X, and I also have a 3X available in white.
tokens: ['adorable', 'top', 'hint', 'lace', 'key', 'hole', 'back', 'pale', 'pink', 'also'
, 'available', 'white']
description: New with tags. Leather horses. Retail for [rm] each. Stand about a foot high
. They are being sold as a pair. Any questions please ask. Free shipping. Just got out of
storage
tokens: ['new', 'tags', 'leather', 'horses', 'retail', 'stand', 'foot', 'high', 'sold', '
pair', 'questions', 'please', 'ask', 'free', 'shipping', 'got', 'storage']
description: Complete with certificate of authenticity
tokens: ['complete', 'certificate', 'authenticity']
```

In [ ]:

```
print(5)
```

```
5
```

In [ ]:

```
```

Tokenizing the preproccesed words

In [ ]:

```
train['tokens'] = train['desc'].map(tokenize)
```

In [ ]:

```
print(5)
```

In [ ]:

```
for description, tokens in zip(train['desc'].head(15),
                               train['tokens'].head(15)):
    print('description:', description)
    print('tokens:', tokens)
    print()
```

```
description: b'descript yet'
tokens: None

description: b'keyboard great condit work like came box port test work perfect light cust
omiz via razer synaps app'
tokens: None

description: b'ador top hint lace key hole back pale pink also avail white'
tokens: None

description: b'new tag leather hors retail stand foot high sold pair question pleas ask f
ree ship got storag'
tokens: None

description: b'complet certif authent'
tokens: None

description: b'banana republ bottom candi skirt match blazer ami byer suit loft bottom ca
mi top'
tokens: None

description: b'size small strap slight shorten fit besid perfect condit'
tokens: None

description: b'get three pair sophi cheer short size small medium girl two sport bra boy
short spandex match set small medium girl item total retail store take today less price o
ne item store'
tokens: None

description: b'girl size small plus green three short total'
tokens: None
```

```
description: b'realiz pant backward pictur dirti hand wash stuf bodi paint porcelain head
hand feet back clown scari tall chip crack minor paint loss place clown circus doll colle
ct'
tokens: None

description: b'full size sephora'
tokens: None

description: b'new pink bodi mist fresh clean sun kiss cool bright total flirt sweet flir
ti'
tokens: None

description: b'great condit'
tokens: None

description: b'descript yet'
tokens: None

description: b'authent sued fring boot great condit size size love wear thick sock winter
theyd perfect well last winter'
tokens: None
```

In [ ]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,
                             max_features=180000,
                             tokenizer=tokenize,
                             ngram_range=(1, 2))
```

In [ ]:

```
vz=vectorizer.fit_transform(train["desc"].apply(str))
```

In [ ]:

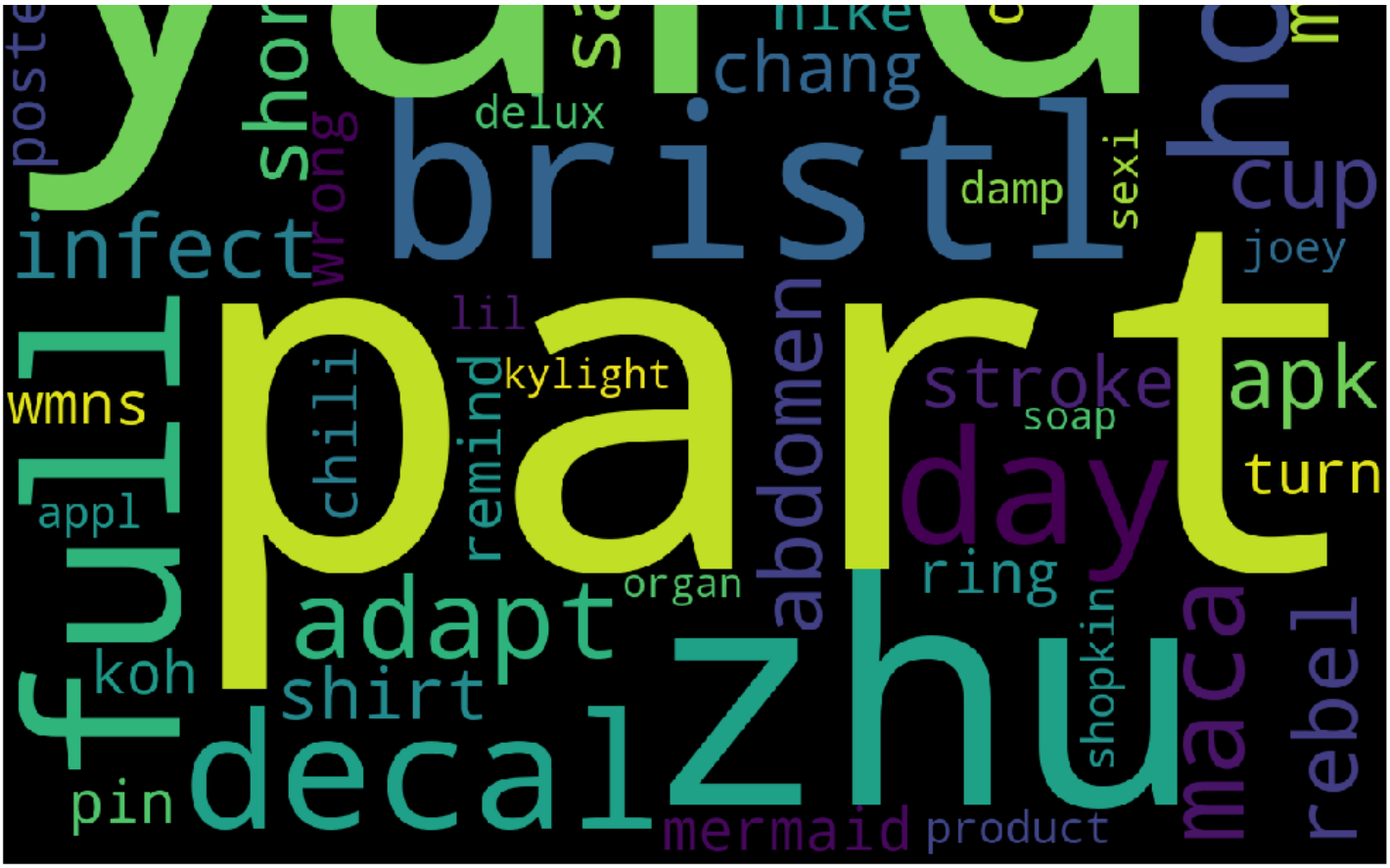
```
all_features2=vectorizer.get_feature_names()
words2=[]
idf2=vectorizer.idf_
features=np.argsort(idf2)[::-1]
for i in features[0:30]:
    words2.append(all_features2[i])
print(words2)
```

```
['zhu', 'bristl full', 'decal home', 'day effect', 'adapt screw', 'maca', 'infect part',
'lower abdomen', 'sanit cup', 'apk', 'rebel maxen', 'short stroke', 'shirt chang', 'dream
mermaid', 'yard yard', 'record poster', 'ring remind', 'rubber pin', 'koh', 'wrong turn',
'dip chili', 'nike wmns', 'product shopkin', 'sexi part', 'slight damp', 'lil joey', 'app
l smartwatch', 'delux tarteist', 'kylight singl', 'organ soap']
```

In [ ]:

```
from wordcloud import WordCloud

wordcloud = WordCloud(width = 1200, height = 1000).generate(" ".join(words2))
plt.figure(figsize = (20, 15))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```



In [ ]:

```
#from sklearn.cluster import KMeans
```

In [ ]:

```
'''from sklearn.cluster import MiniBatchKMeans

num_clusters = 30 # need to be selected wisely
kmeans_model = MiniBatchKMeans(n_clusters=num_clusters,
                               init='k-means++',
                               n_init=1,
                               init_size=1000, batch_size=1000, verbose=0, max_iter=1000
)'''
```

Out[ ]:

```
"from sklearn.cluster import MiniBatchKMeans\n\nnum_clusters = 30 # need to be selected w
isely\nkmeans_model = MiniBatchKMeans(n_clusters=num_clusters,\n
init='k-means++',\n                                    n_init=1,\n
init_size=1000, batch_size=1000, verbose=0, max_iter=1000)"
```

In [ ]:

```

```

In [ ]:

```
'''kmeans = kmeans_model.fit(vectorizer)
kmeans_clusters = kmeans.predict(vectorizer)
kmeans_distances = kmeans.transform(vectorizer)'''
```

Out[ ]:

```
'kmeans = kmeans_model.fit(vectorizer)\nkmeans_clusters = kmeans.predict(vectorizer)\nkme
ans_distances = kmeans.transform(vectorizer)'
```

In [ ]:

```
'''optimal_k = KMeans(n_clusters = 3)
p = optimal_k.fit(vz)'''
```

Out[ ]:

Out[ ]:

```
'optimal_k = KMeans(n_clusters = 3)\np = optimal_k.fit(vz)'
```

In [ ]:

```
'''cluster1,cluster2,cluster3=[],[],[]
for i in range(p.labels_.shape[0]):
    if clf.labels_[i] == 0:
        cluster1.append(train['desc'][i])
data=''
for i in cluster1:
    data+=str(i)

from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(data)
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()'''
```

Out[ ]:

```
'cluster1,cluster2,cluster3=[],[],[]\nfor i in range(p.labels_.shape[0]):\n    if clf.lab
els_[i] == 0:\n        cluster1.append(train[\'desc\'][i])\ndata=\'\'\nfor i in cluster1:
\n    data+=str(i)\n\nfrom wordcloud import WordCloud\nwordcloud = WordCloud(background_c
olor="white").generate(data)\n# Display the generated image:\nplt.imshow(wordcloud, inter
polation=\'bilinear\')\nplt.axis("off")\nplt.show()'
```

In [ ]:

In [ ]:

Shipping Feature

In [ ]:

```
train.shipping.value_counts()
```

Out[ ]:

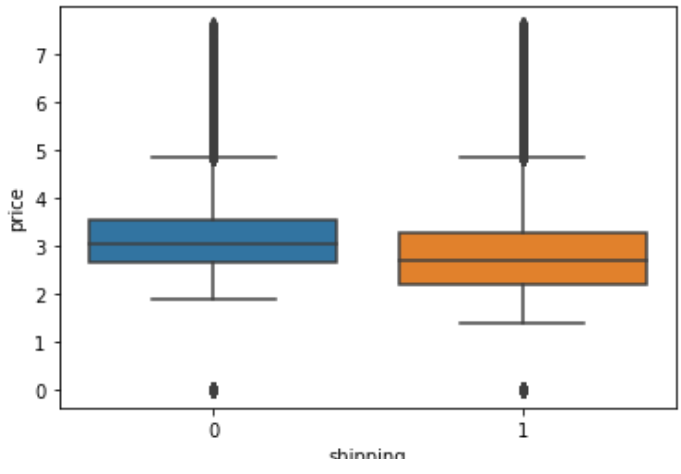
```
0    819435
1    663100
Name: shipping, dtype: int64
```

In [ ]:

```
sns.boxplot(x = train.shipping, y = log_price, orient = "v")
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x254e4685e48>
```



In [ ]:

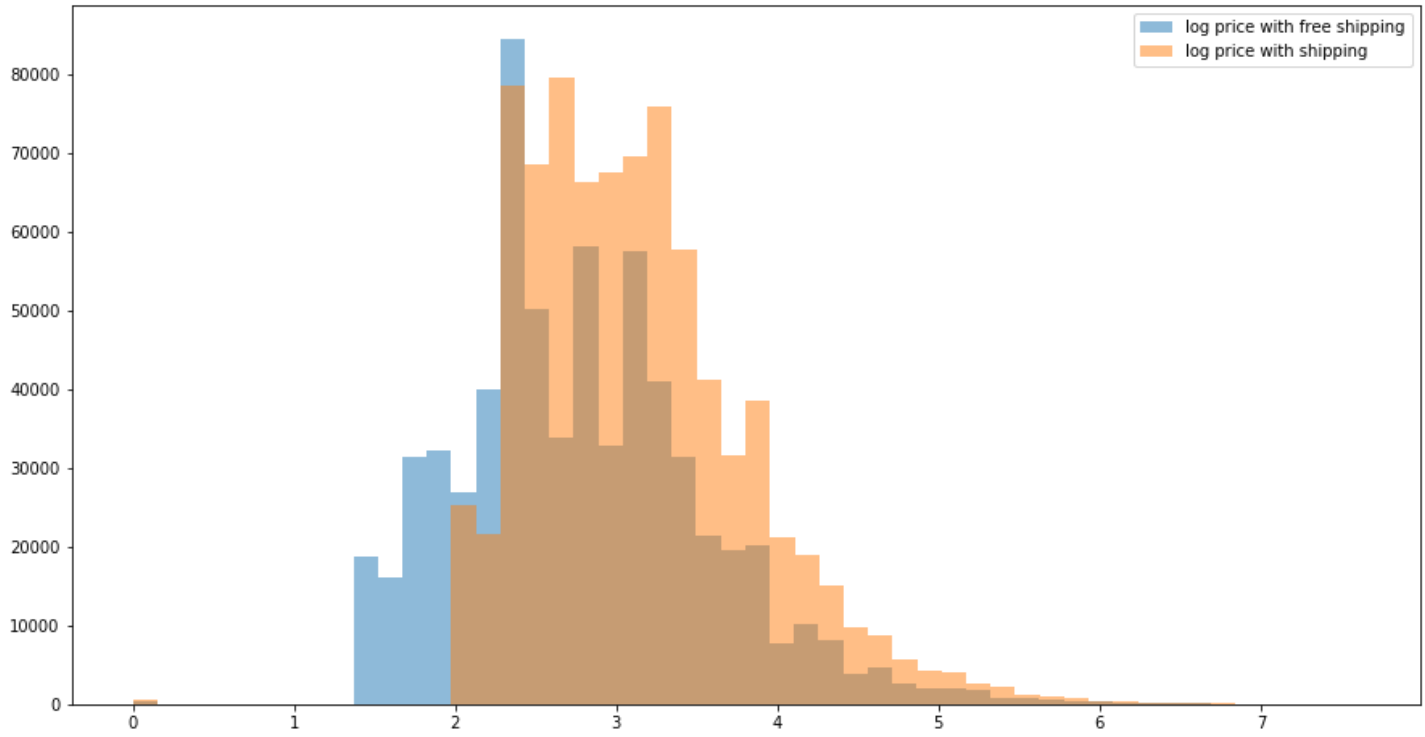
```
```

In [ ]:

```
train['log_price']=log_price
```

In [ ]:

```
plt.figure(figsize = (15, 8))
plt.hist(train[train.shipping == 1].log_price, bins = 50, alpha = 0.5, label = "log price with free shipping")
plt.hist(train[train.shipping == 0].log_price, bins = 50, alpha = 0.5, label = "log price with shipping")
plt.legend(fontsize = 10)
plt.show()
```



In [ ]:

```
```

In [ ]:

```
```

Item condition of normal points

In [ ]:

```
def nol(data, m = 2):
    return data[abs(data - np.mean(data)) < m * np.std(data)]
```

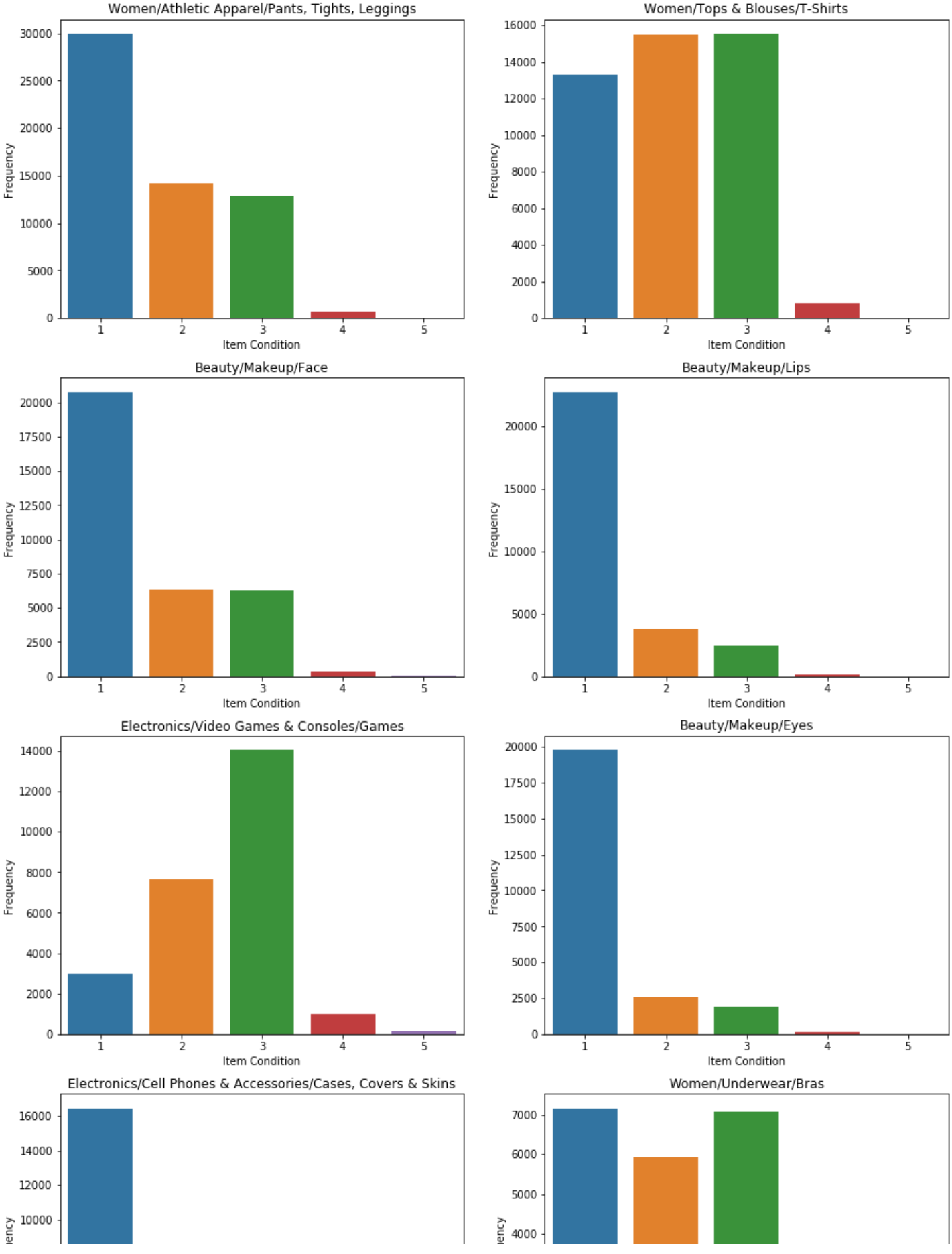
In [ ]:

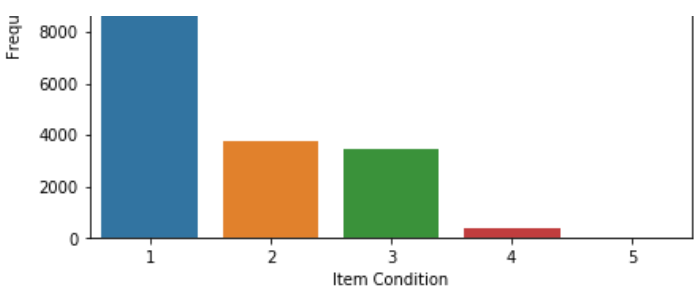
```
chist = train.groupby(["category_name"], as_index = False).count().sort_values(by = "train_id",
                                                                                ascending = False)[0:25]
```

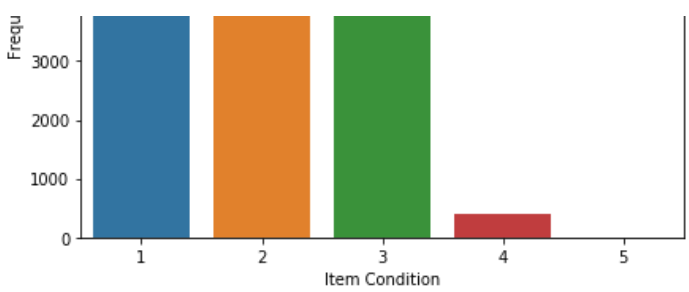
In [ ]:

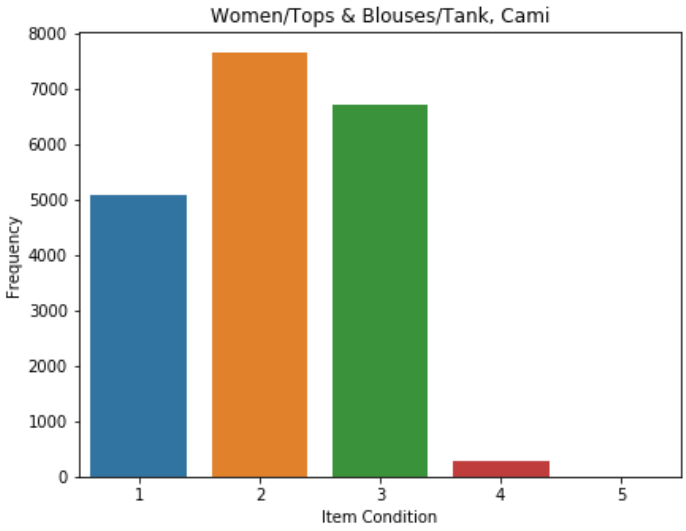
```
```

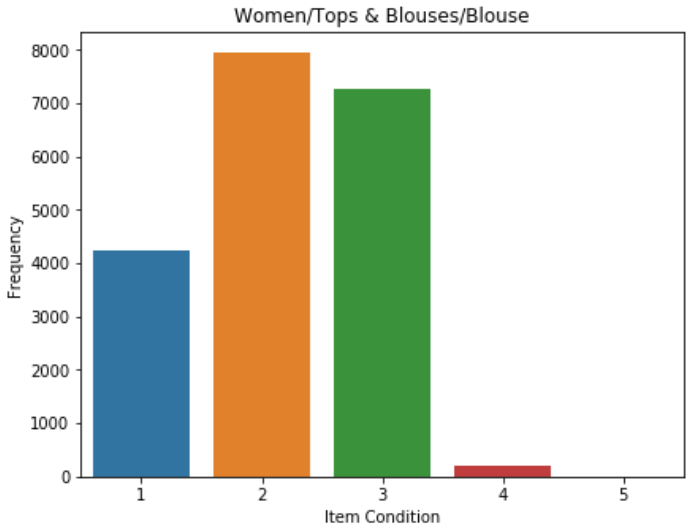
```
fig, ax = plt.subplots(5, 2, figsize = (15, 30))
k=10
for i in range(k):
    ohist = train.iloc[(nol(train[
        train["category_name"] == chist["category_name"].values[i]
    ].price).index).values].groupby(["item_condition_id"], as_index = False).count()
    sns.barplot(x = ohist["item_condition_id"], y = ohist["train_id"], ax = ax[int(i/2)]
[i%2])
    ax[int(i/2)][i%2].set_title(chist["category_name"].values[i])
    ax[int(i/2)][i%2].set_xlabel("Item Condition")
    ax[int(i/2)][i%2].set_ylabel("Frequency")
```











In [ ]:

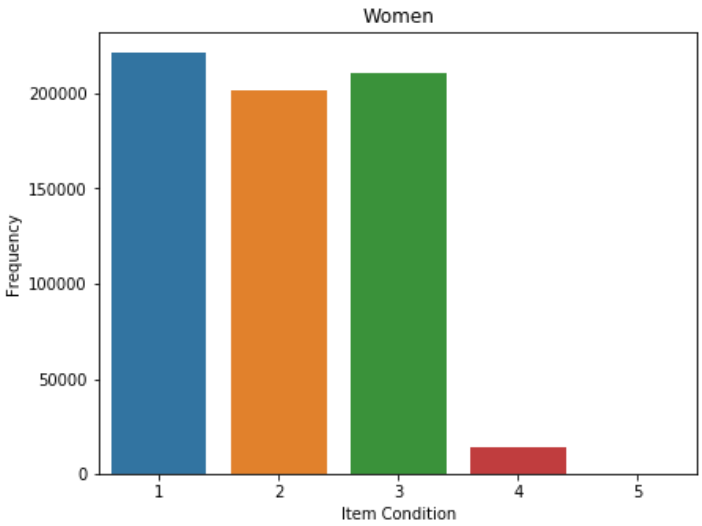
```
```

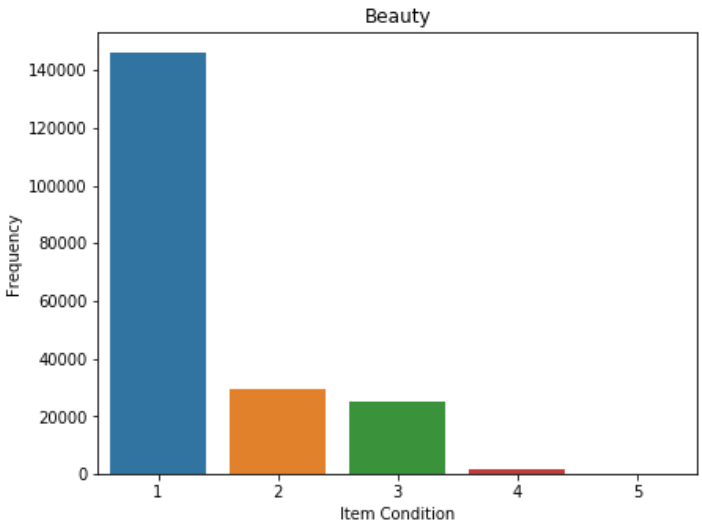
In [ ]:

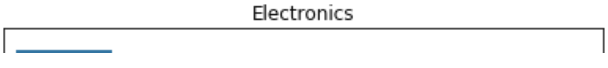
```
chist1 = train.groupby(["general_cat"], as_index = False).count().sort_values(by = "trai
n_id",
                                                                               ascendin
g = False)[0:25]
```

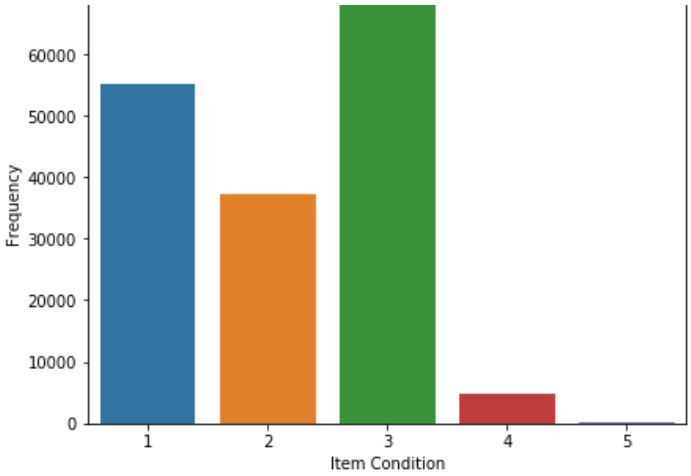
In [ ]:

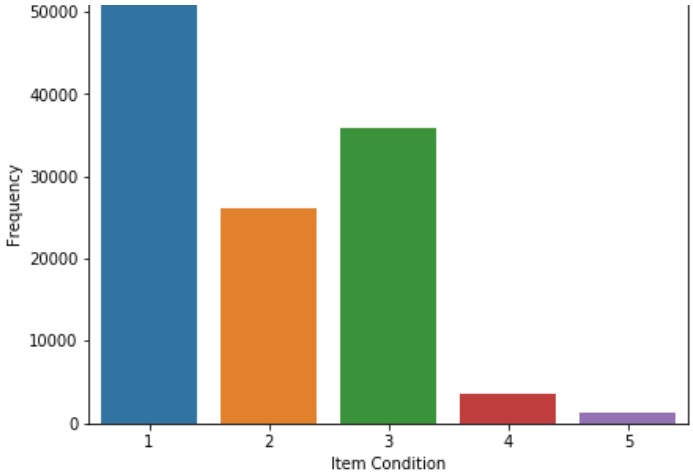
```
fig, ax = plt.subplots(5, 2, figsize = (15, 30))
k=10
for i in range(k):
    ohist = train.iloc[(nol(train[
        train["general_cat"] == chist1["general_cat"].values[i]
    ].price).index).values].groupby(["item_condition_id"], as_index = False).count()
    sns.barplot(x = ohist["item_condition_id"], y = ohist["train_id"], ax = ax[int(i/2)]
[i%2])
    ax[int(i/2)][i%2].set_title(chist1["general_cat"].values[i])
    ax[int(i/2)][i%2].set_xlabel("Item Condition")
    ax[int(i/2)][i%2].set_ylabel("Frequency")
```

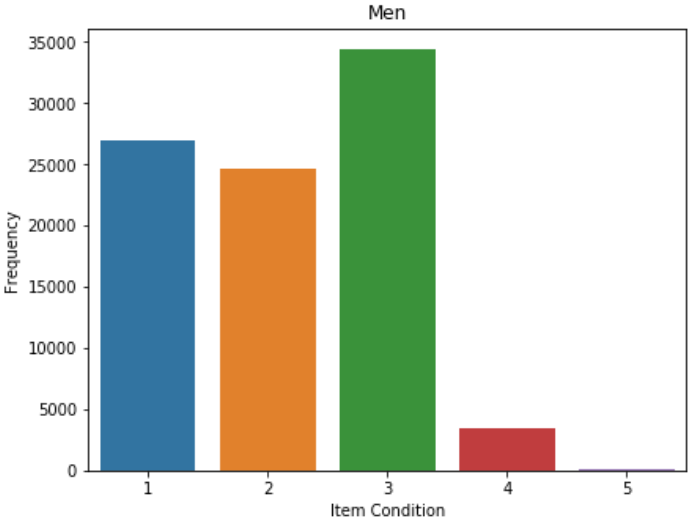


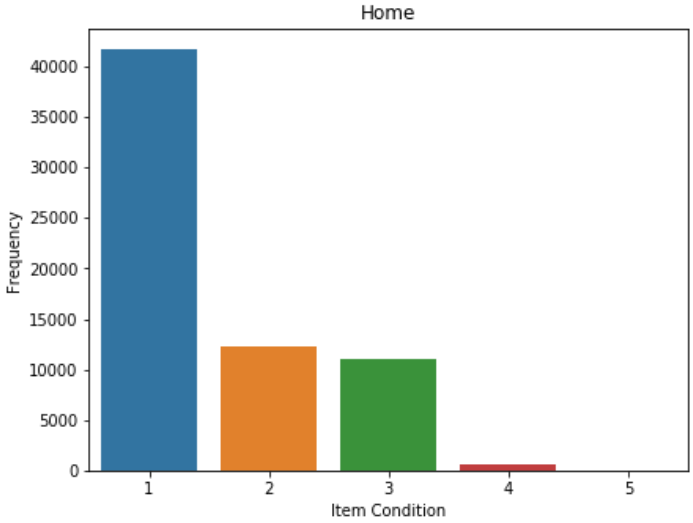


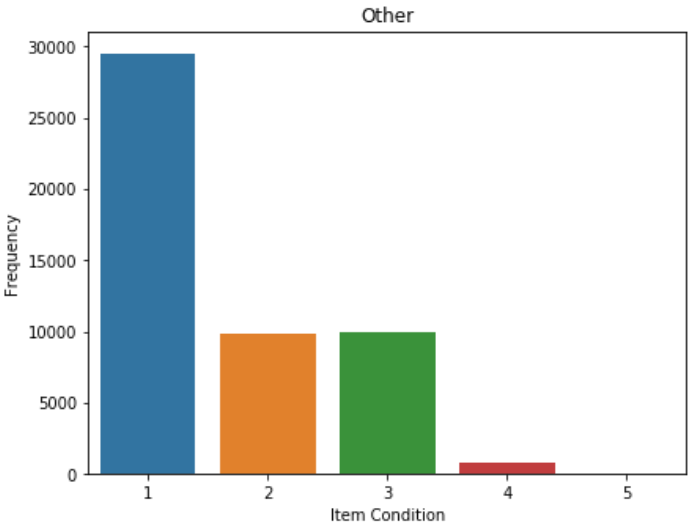


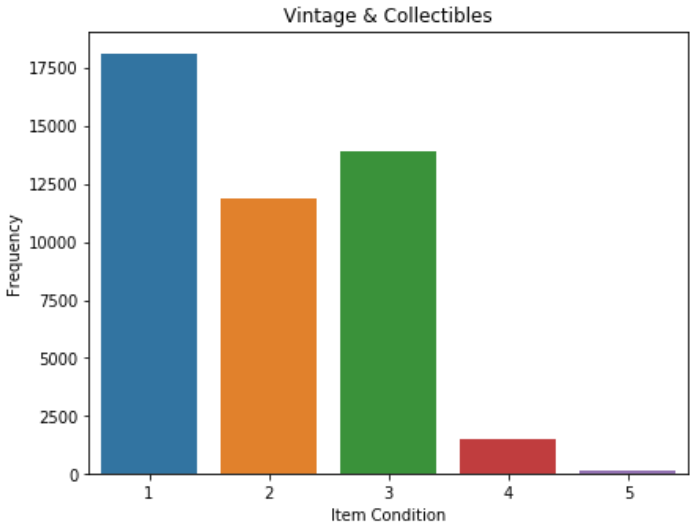

Frequency
Item Condition

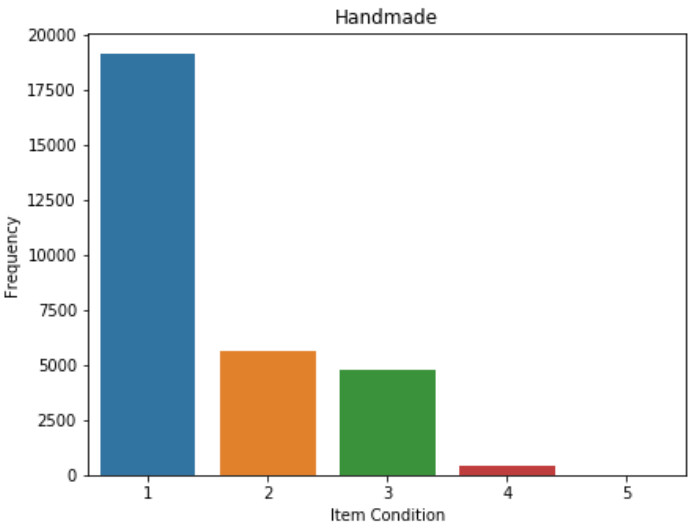

Frequency
Item Condition

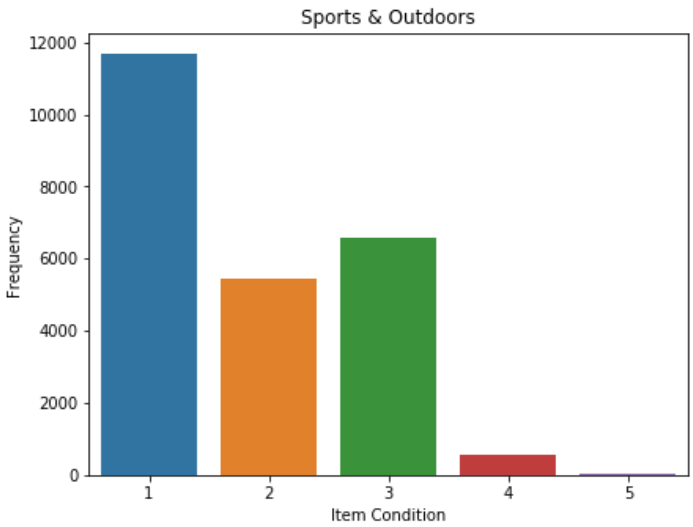

Men
Frequency
Item Condition


Home
Frequency
Item Condition


Other
Frequency
Item Condition


Vintage & Collectibles
Frequency
Item Condition


Handmade
Frequency
Item Condition


Sports & Outdoors
Frequency
Item Condition

In [ ]:

In [ ]:

```
```

In [ ]:

```
```

Item condition of outlier points

In [ ]:

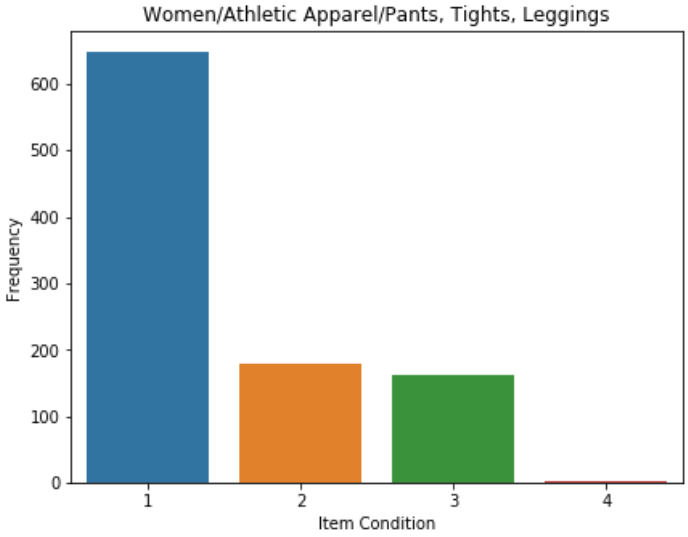
```
# outliers
def ol(data, m = 3):
    return data[(data - np.mean(data)) >= m * np.std(data)]
```

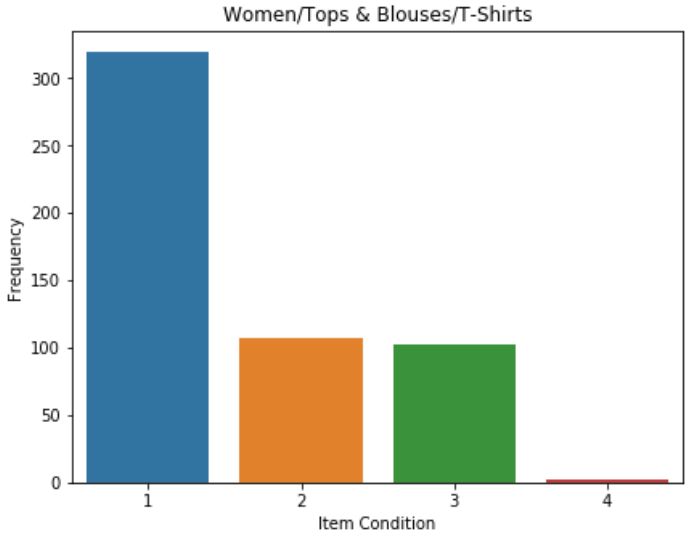
In [ ]:

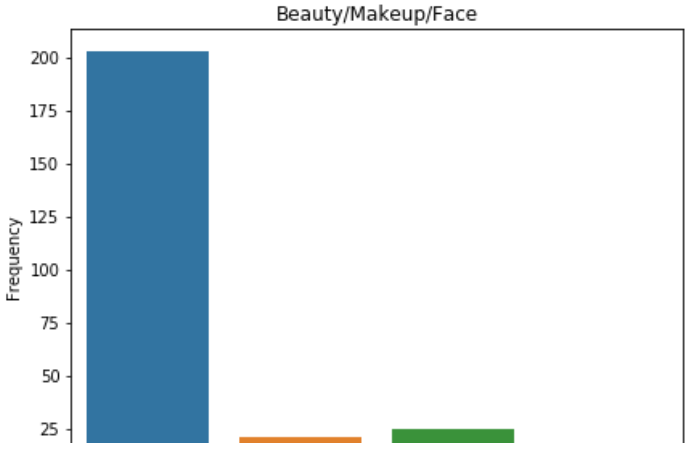
```
chist = train.groupby(["category_name"], as_index = False).count().sort_values(by = "tra
in_id",
                                                                                ascendin
g = False)[0:25]
```

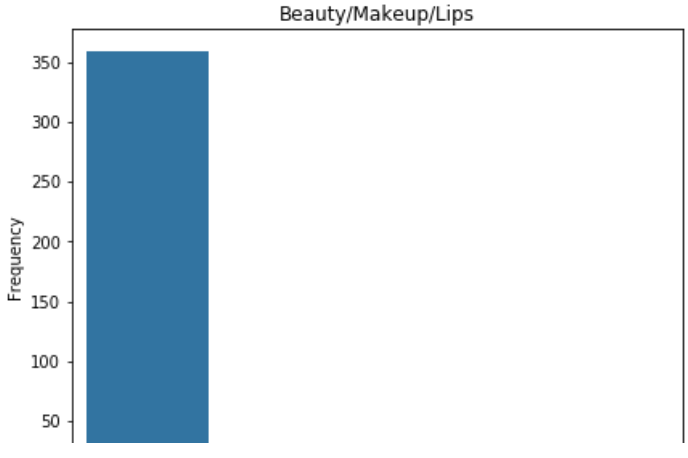
In [ ]:

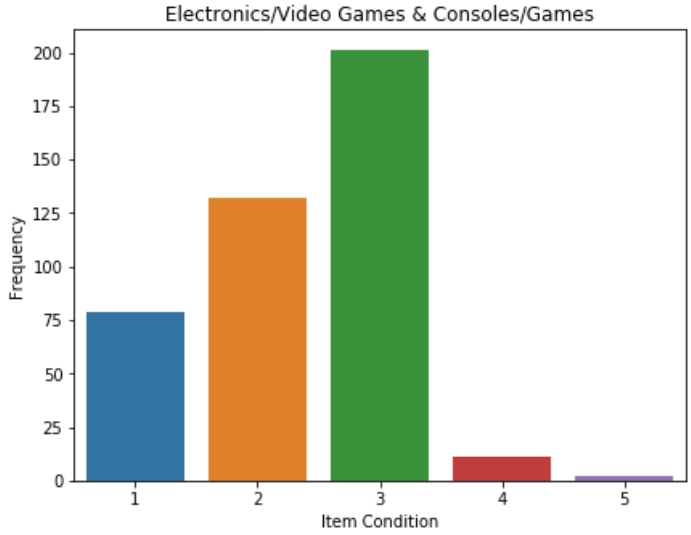
```
fig, ax = plt.subplots(5, 2, figsize = (15, 30))
k=10
for i in range(k):
    ohist = train.iloc[(ol(train[
        train["category_name"] == chist["category_name"].values[i]
    ].price).index).values].groupby(["item_condition_id"], as_index = False).count()
    sns.barplot(x = ohist["item_condition_id"], y = ohist["train_id"], ax = ax[int(i/2)]
[i%2])
    ax[int(i/2)][i%2].set_title(chist["category_name"].values[i])
    ax[int(i/2)][i%2].set_xlabel("Item Condition")
    ax[int(i/2)][i%2].set_ylabel("Frequency")
```

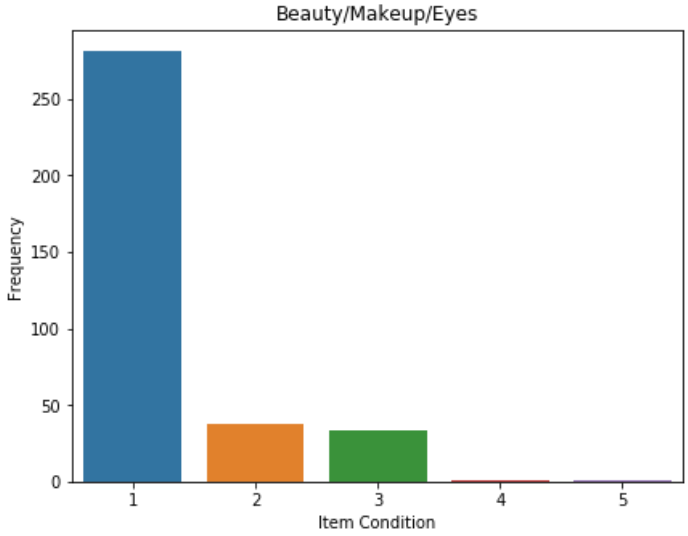


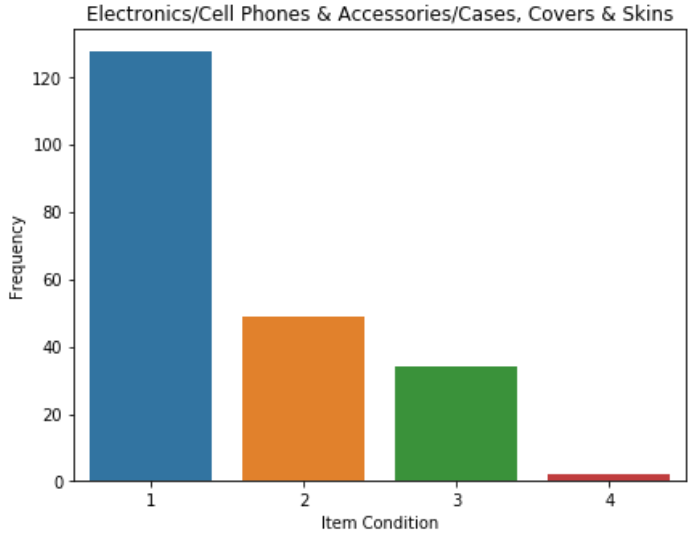


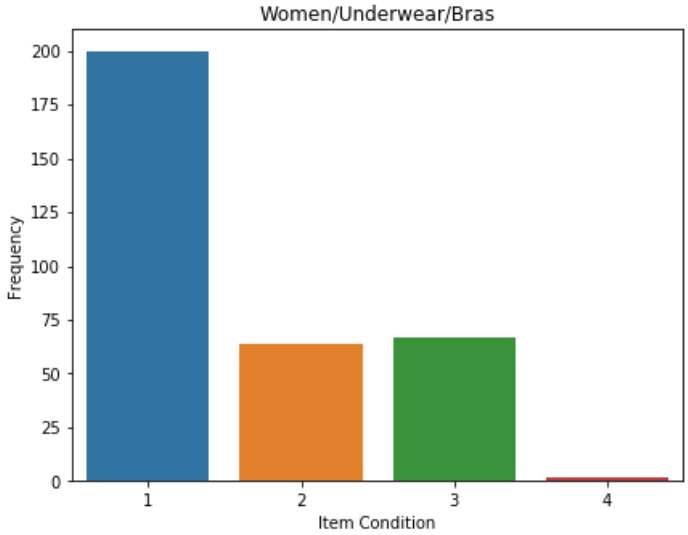


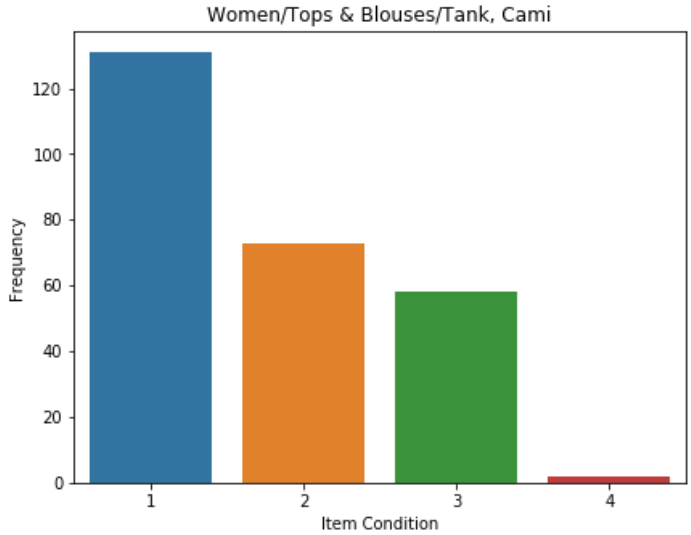


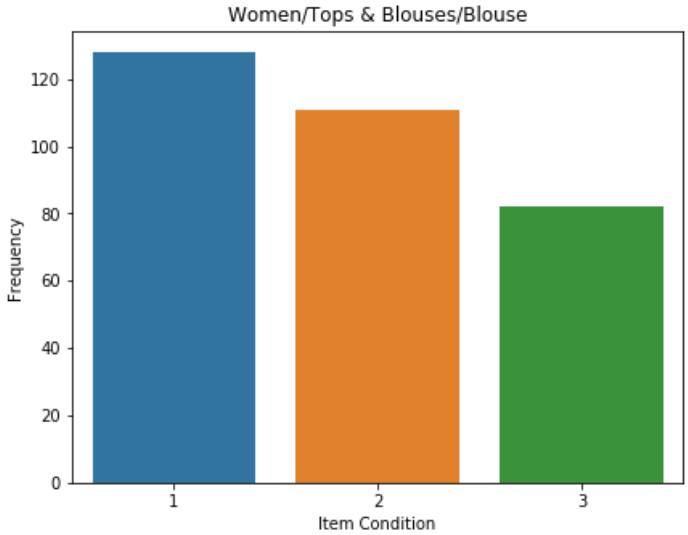












In [ ]:

```
```

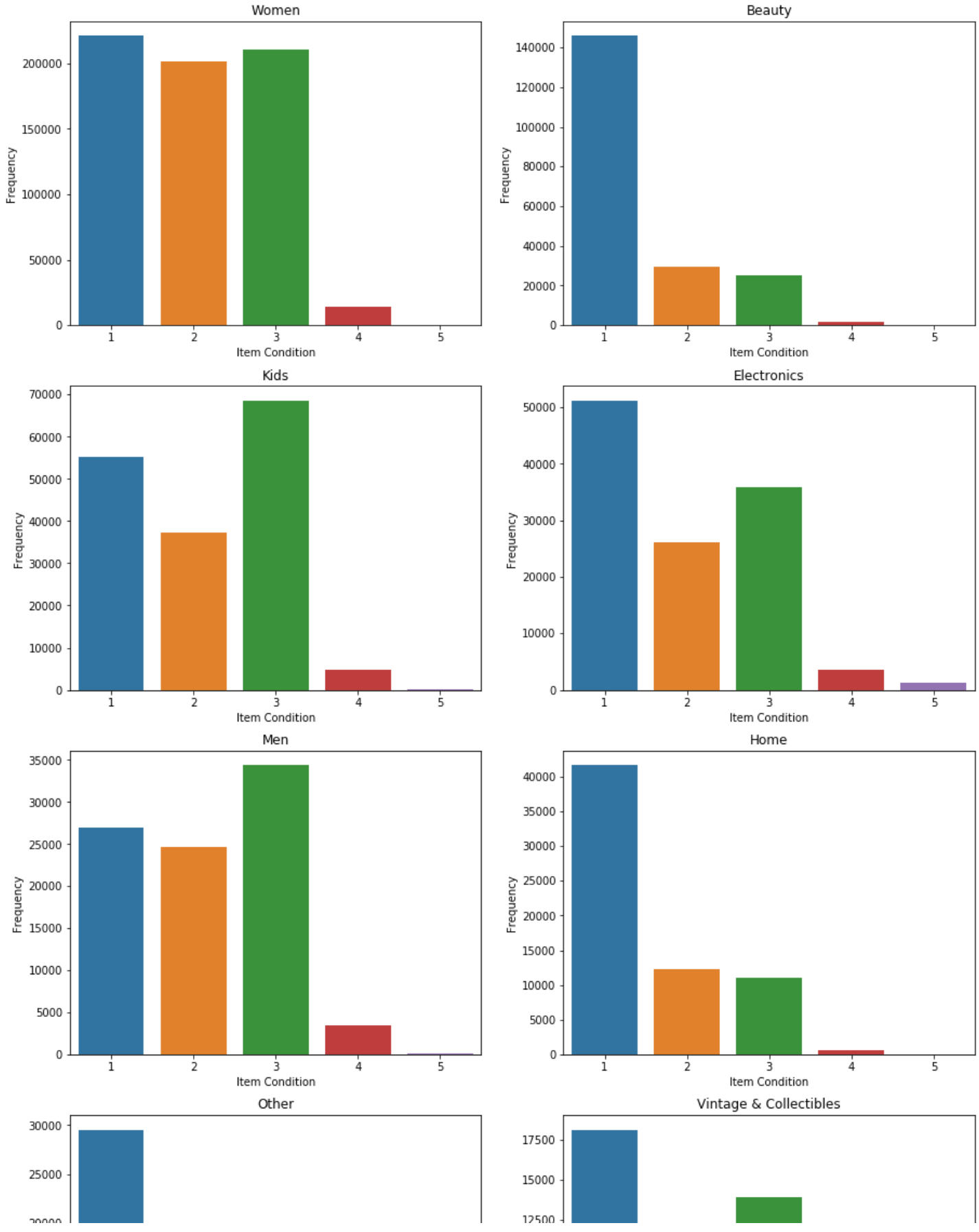
General Categories vs Item Condition

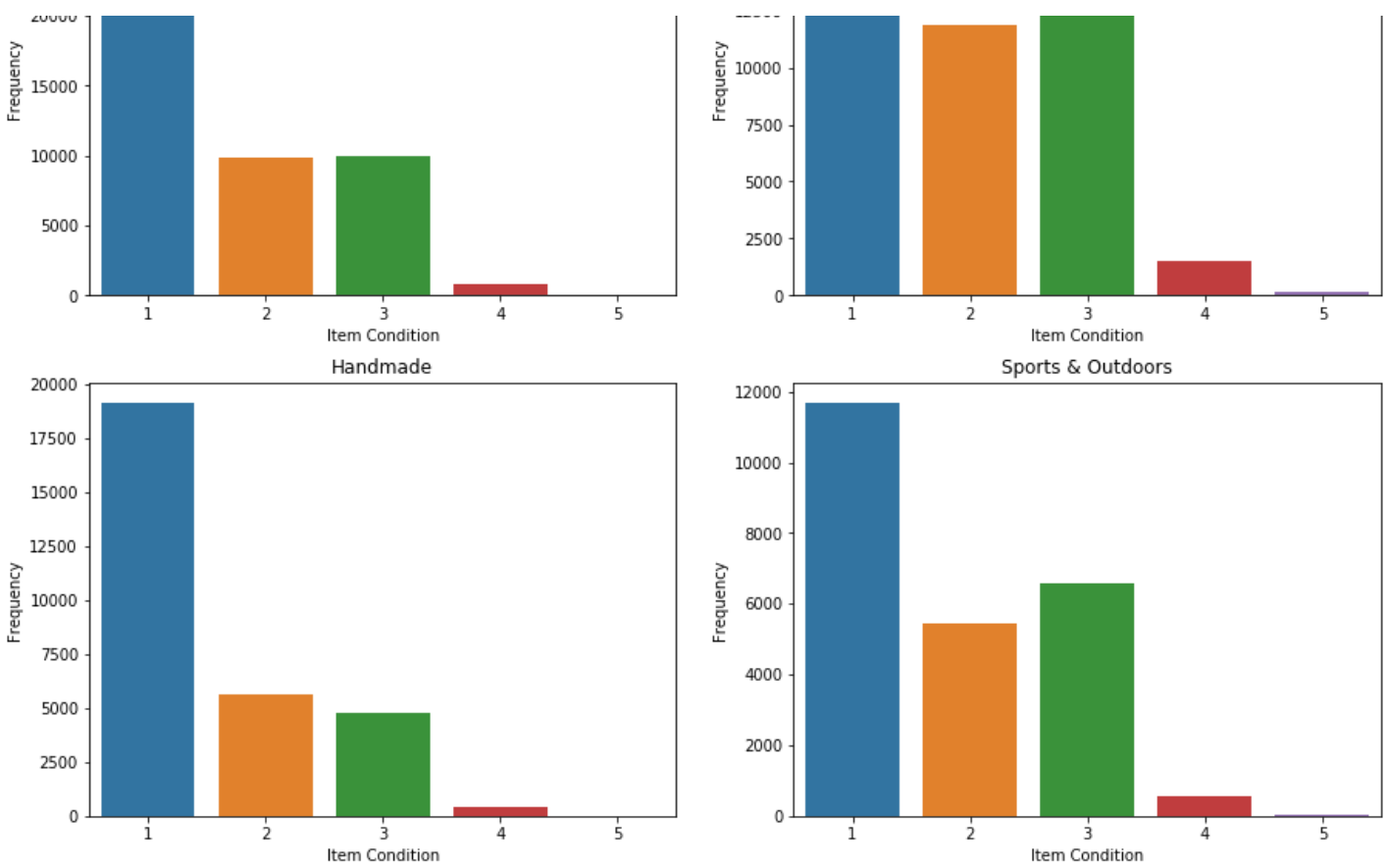
In [ ]:

```
chist1 = train.groupby(["general_cat"], as_index = False).count().sort_values(by = "trai
n_id",
                                                                               ascendin
g = False)[0:25]
```

In [ ]:

```
fig, ax = plt.subplots(5, 2, figsize = (15, 30))
k=10
for i in range(k):
    ohist = train.iloc[(nol(train[
        train["general_cat"] == chist1["general_cat"].values[i]
    ].price).index).values].groupby(["item_condition_id"], as_index = False).count()
    sns.barplot(x = ohist["item_condition_id"], y = ohist["train_id"], ax = ax[int(i/2)]
[i%2])
    ax[int(i/2)][i%2].set_title(chist1["general_cat"].values[i])
    ax[int(i/2)][i%2].set_xlabel("Item Condition")
    ax[int(i/2)][i%2].set_ylabel("Frequency")
```





In [ ]:

In [ ]:

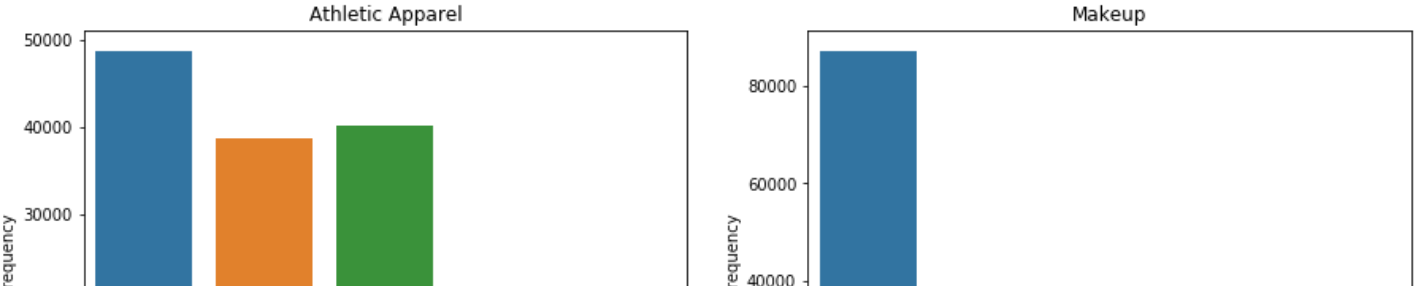
Sub catergories normal points vs Item Condition

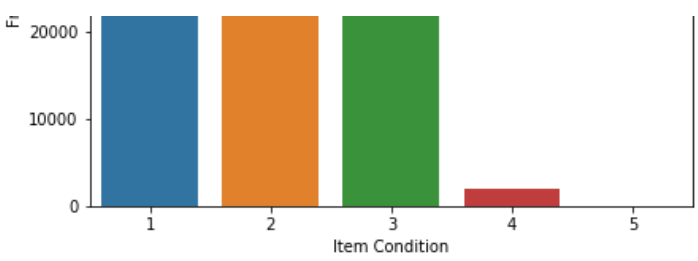
In [ ]:

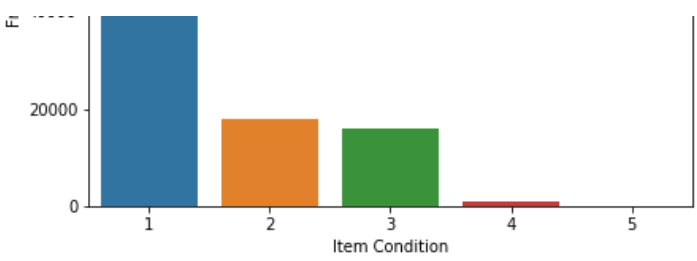
```
chist2 = train.groupby(["subcat_1"], as_index = False).count().sort_values(by = "train_id",
                                                                           ascending = False)[0:25]
```

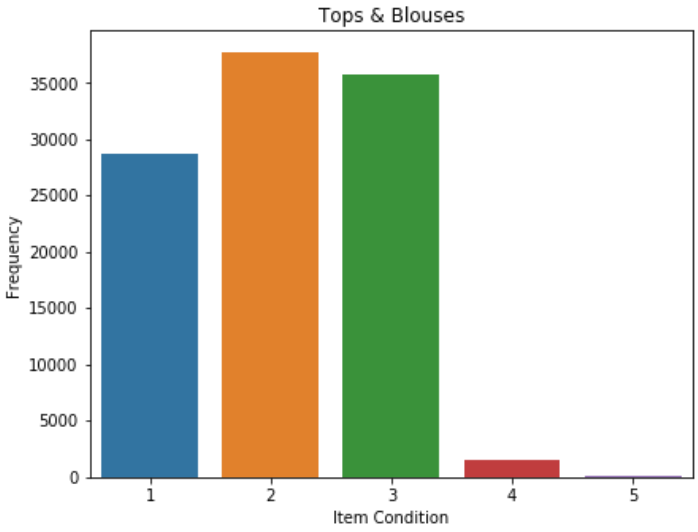
In [ ]:

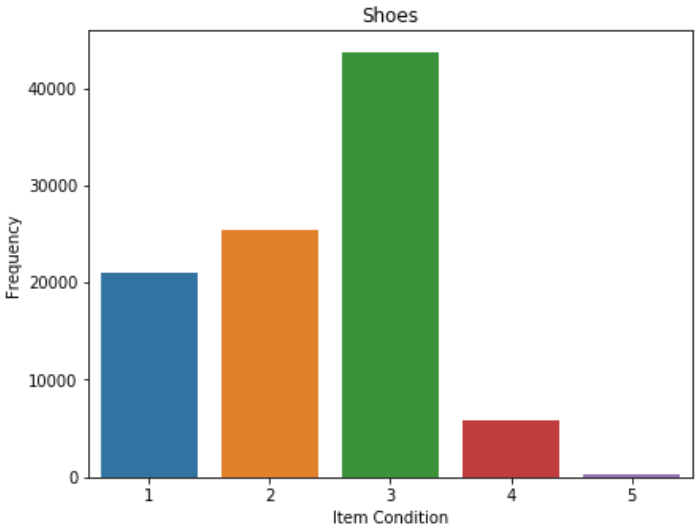
```
fig, ax = plt.subplots(5, 2, figsize = (15, 30))
k=10
for i in range(k):
    ohist = train.iloc[(nol(train[
        train["subcat_1"] == chist2["subcat_1"].values[i]
    ].price).index).values].groupby(["item_condition_id"], as_index = False).count()
    sns.barplot(x = ohist["item_condition_id"], y = ohist["train_id"], ax = ax[int(i/2)][i%2])
    ax[int(i/2)][i%2].set_title(chist2["subcat_1"].values[i])
    ax[int(i/2)][i%2].set_xlabel("Item Condition")
    ax[int(i/2)][i%2].set_ylabel("Frequency")
```

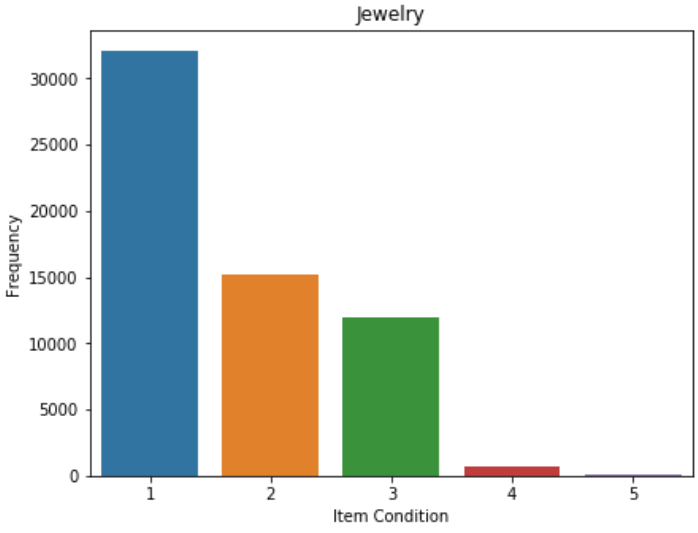


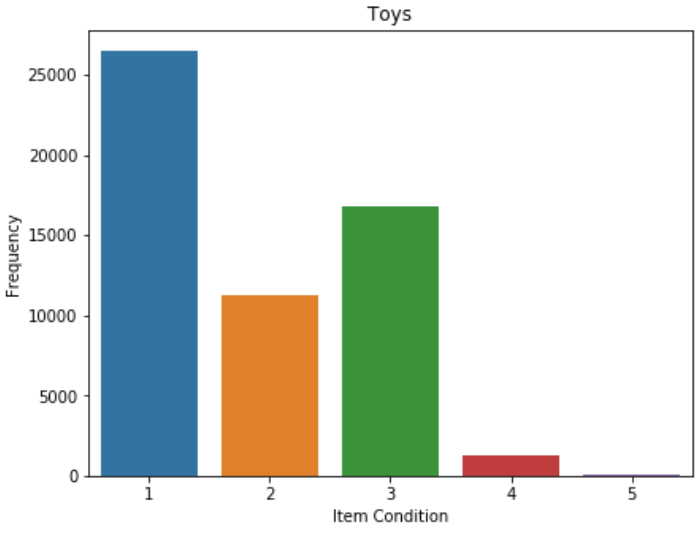

20000
10000
0
1
2
3
4
5
Item Condition

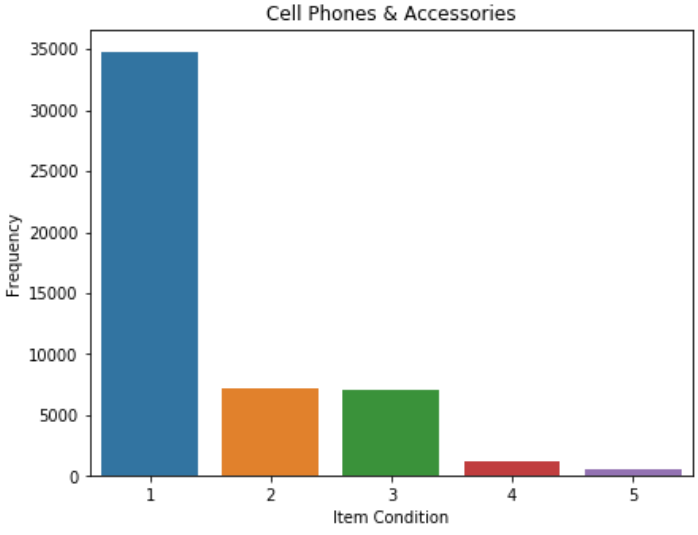

20000
0
1
2
3
4
5
Item Condition

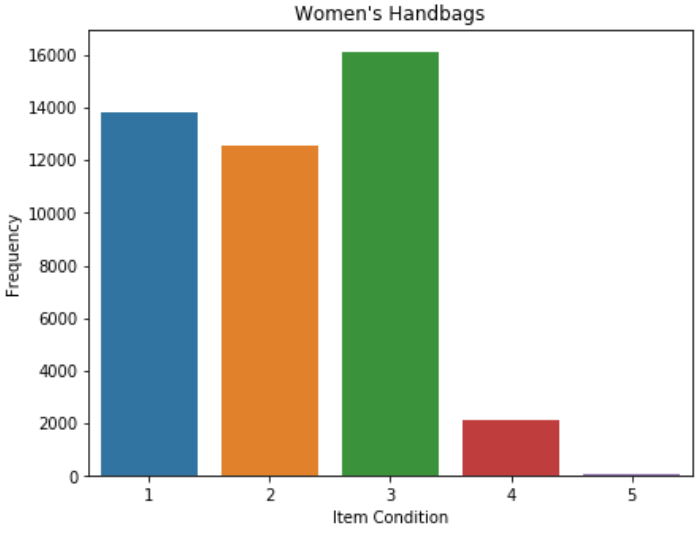

Tops & Blouses
Frequency
35000
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5
Item Condition

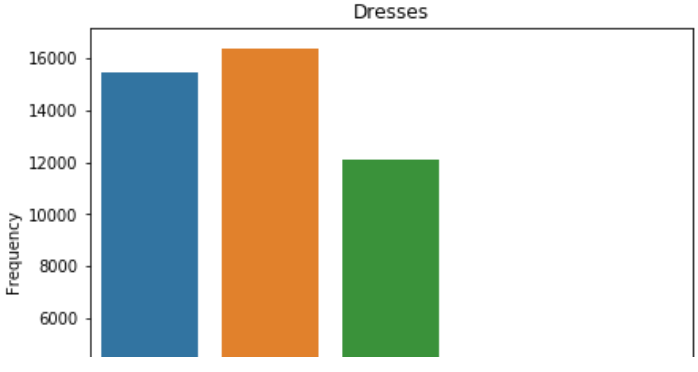

Shoes
Frequency
40000
30000
20000
10000
0
1
2
3
4
5
Item Condition

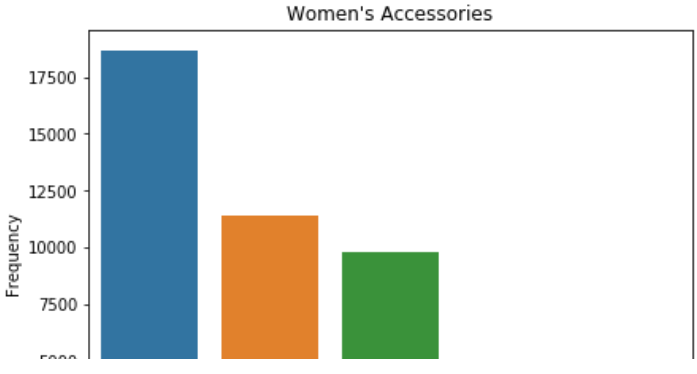

Jewelry
Frequency
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5
Item Condition

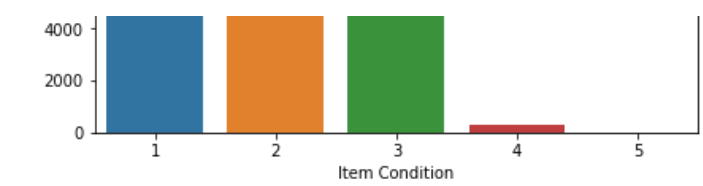

Toys
Frequency
25000
20000
15000
10000
5000
0
1
2
3
4
5
Item Condition

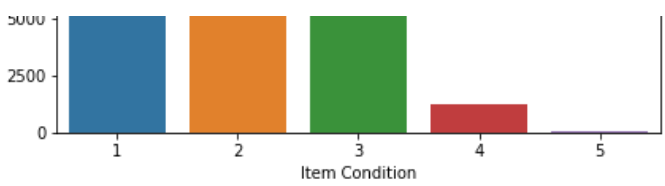

Cell Phones & Accessories
Frequency
35000
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5
Item Condition


Women's Handbags
Frequency
16000
14000
12000
10000
8000
6000
4000
2000
0
1
2
3
4
5
Item Condition


Dresses
Frequency
16000
14000
12000
10000
8000
6000


Women's Accessories
Frequency
17500
15000
12500
10000
7500





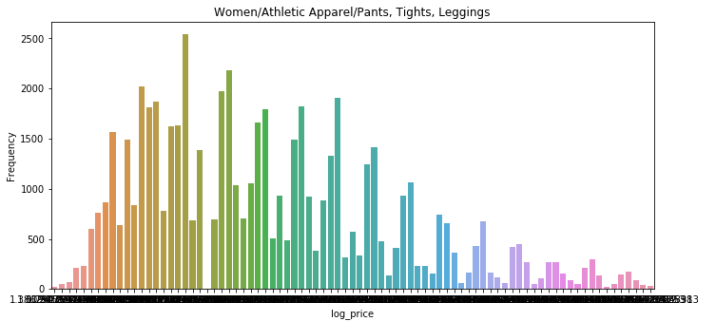
In [ ]:

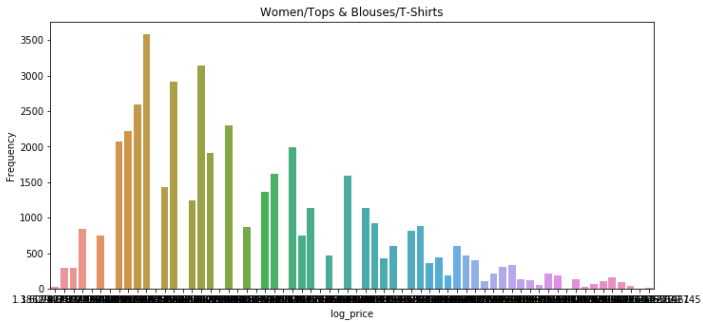
In [ ]:

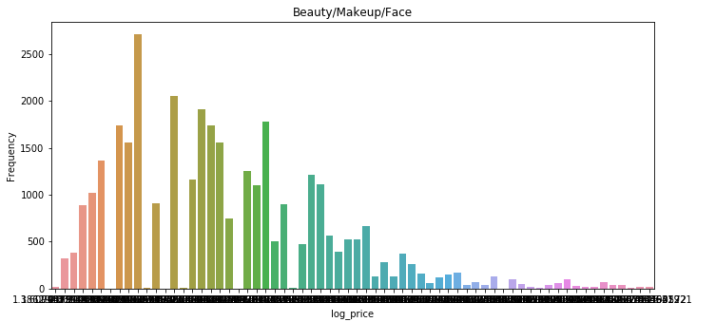
Price vs Frequency of Categories

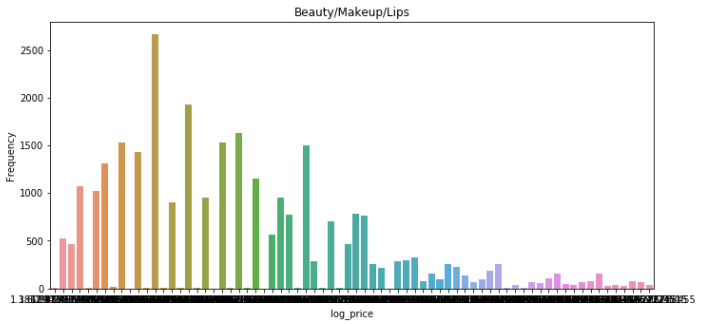
In [ ]:

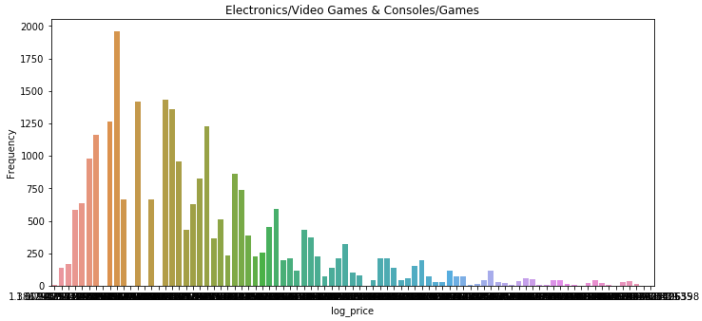
```
fig, ax = plt.subplots(5, 2, figsize = (25, 30))
k=10
for i in range(k):
    ohist = train.iloc[(nol(train[
        train["category_name"] == chist["category_name"].values[i]
    ].price).index).values].groupby(["log_price"], as_index = False).count()
    sns.barplot(x = ohist["log_price"], y = ohist["train_id"], ax = ax[int(i/2)][i%2])
    ax[int(i/2)][i%2].set_title(chist["category_name"].values[i])
    ax[int(i/2)][i%2].set_xlabel("log_price")
    ax[int(i/2)][i%2].set_ylabel("Frequency")
```

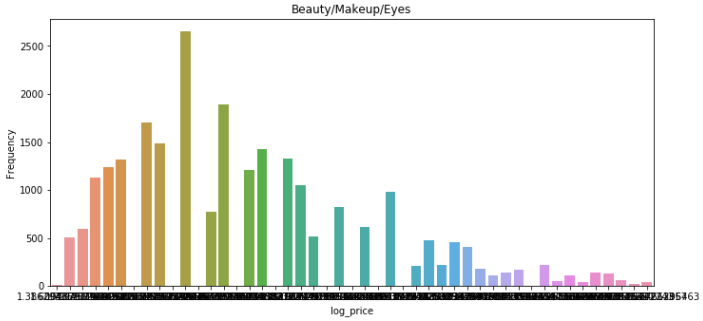


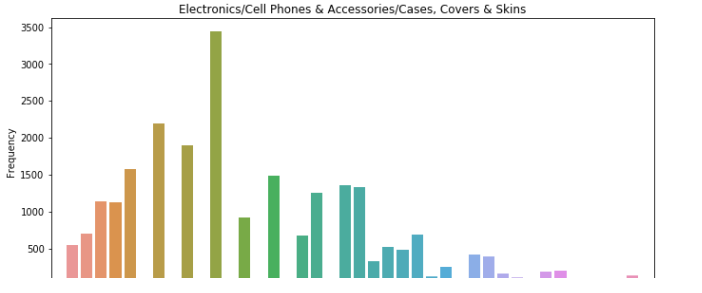


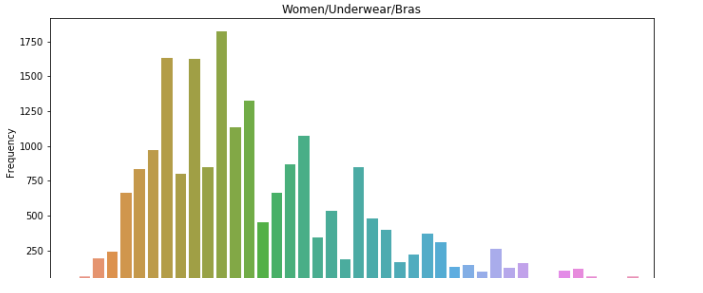


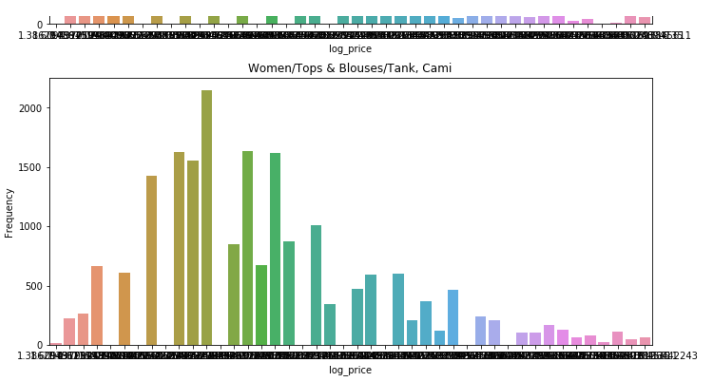


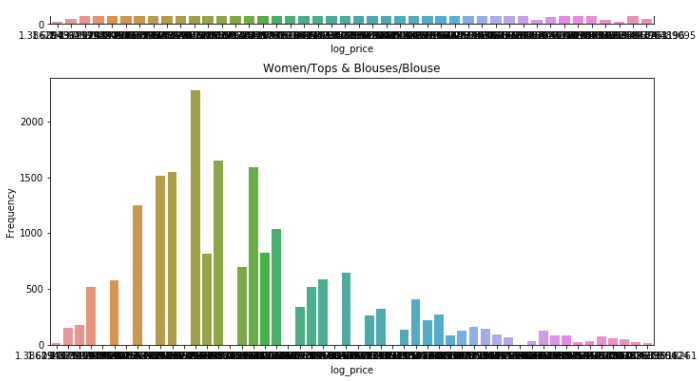












In [ ]:

In [ ]:

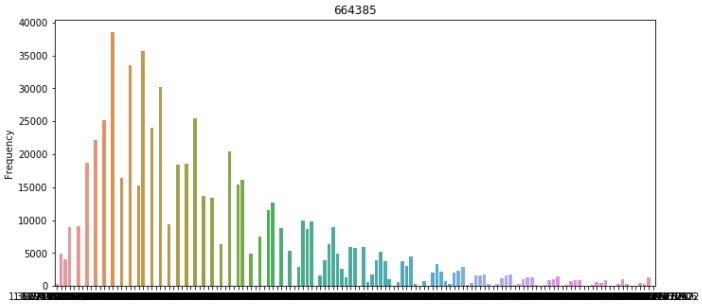
Price vs Frequency of General categories

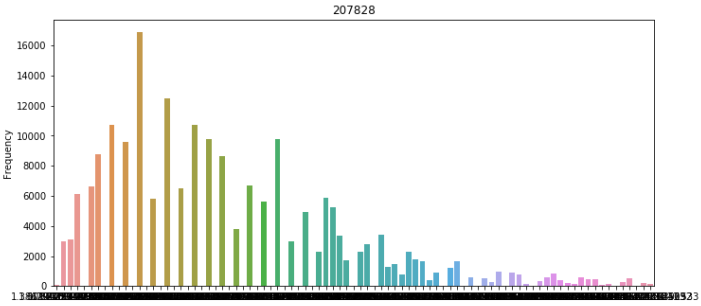
In [ ]:

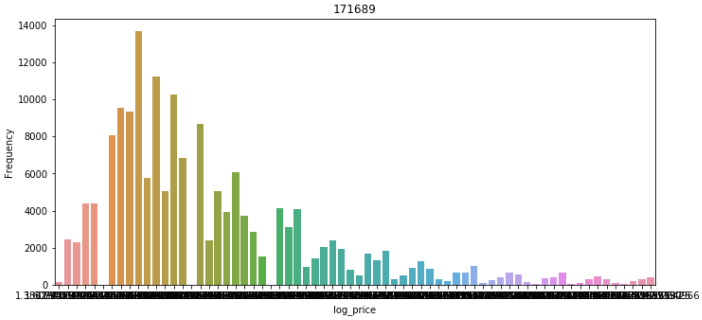
```
chist = train.groupby(["general_cat"], as_index = False).count().sort_values(by = "train
_id", ascending = False)[0:25]
```

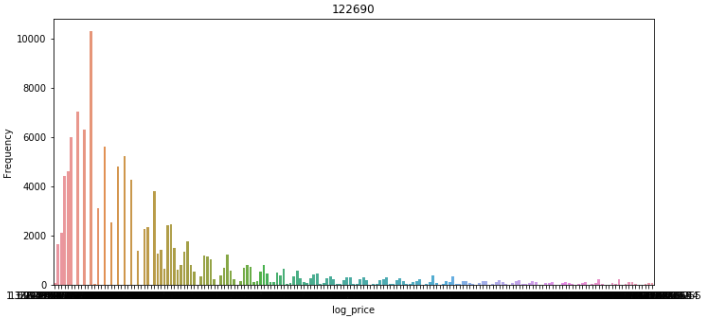
In [ ]:

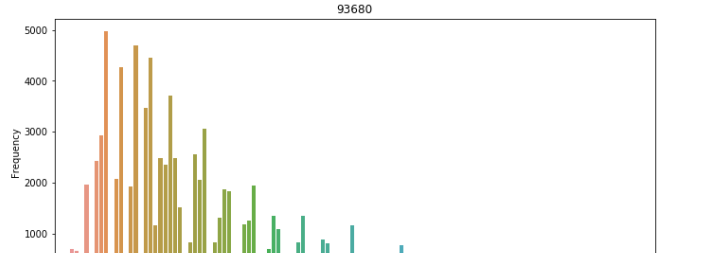
```
fig, ax = plt.subplots(5, 2, figsize = (25, 30))
k=10
for i in range(k):
    ohist = train.iloc[(nol(train[
        train["general_cat"] == chist["general_cat"].values[i]
    ].price).index).values].groupby(["log_price"], as_index = False).count()
    sns.barplot(x = ohist["log_price"], y = ohist["train_id"], ax = ax[int(i/2)][i%2])
    ax[int(i/2)][i%2].set_title(chist["category_name"].values[i])
    ax[int(i/2)][i%2].set_xlabel("log_price")
    ax[int(i/2)][i%2].set_ylabel("Frequency")
```

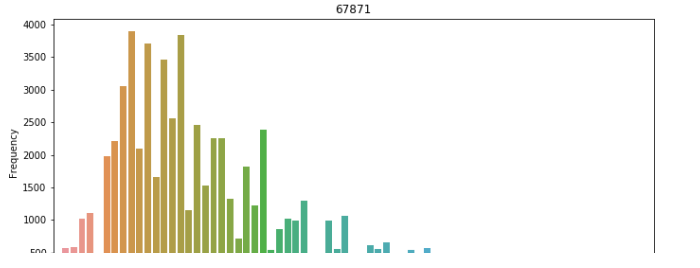


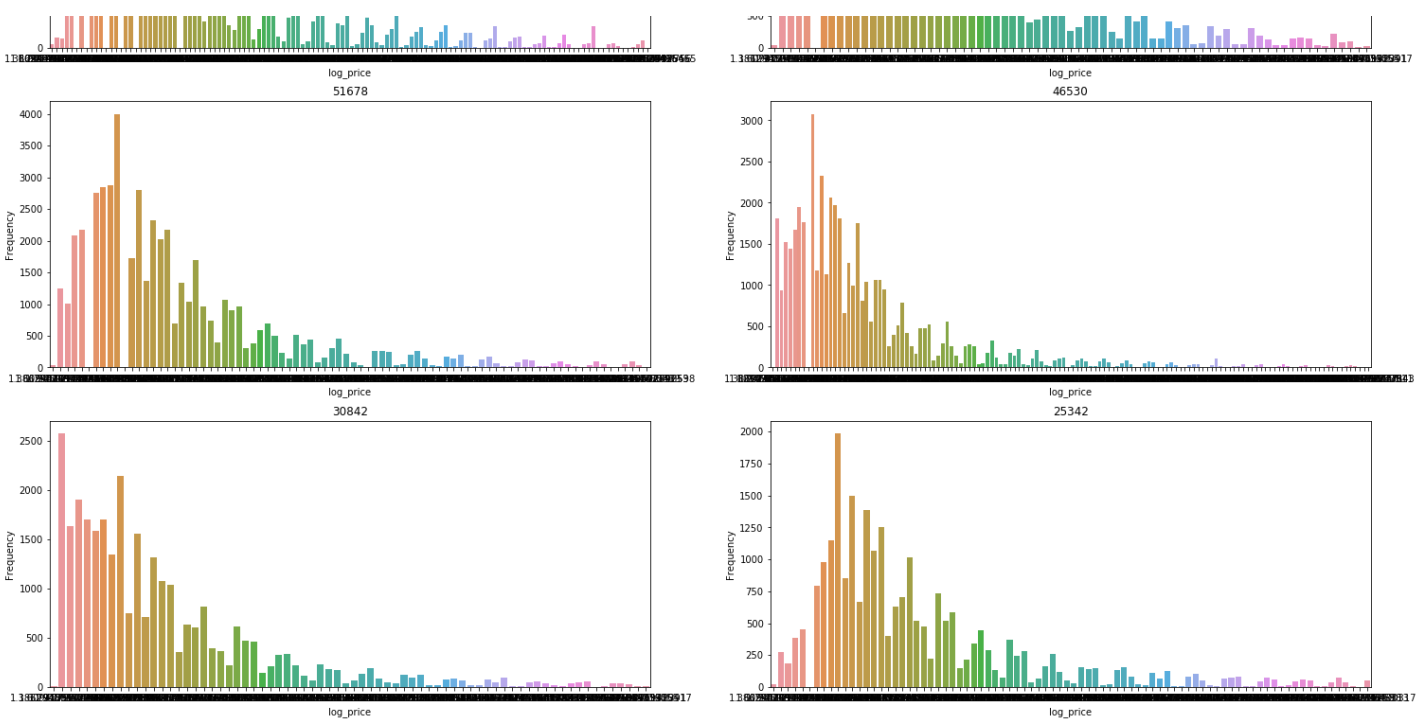












In [ ]:

In [ ]:

In [ ]:

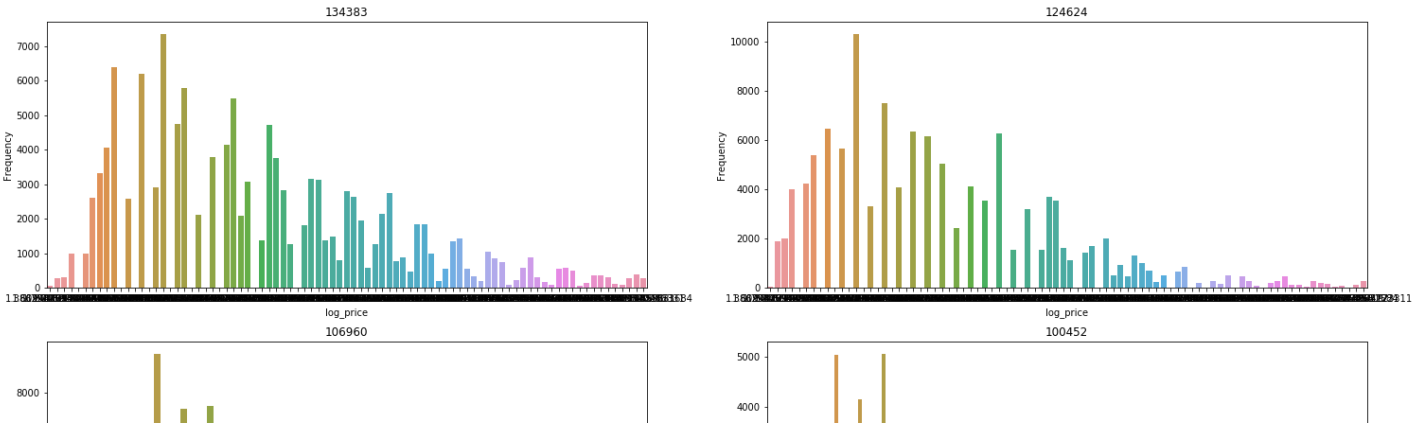
Price vs Frequency of Subcat_1

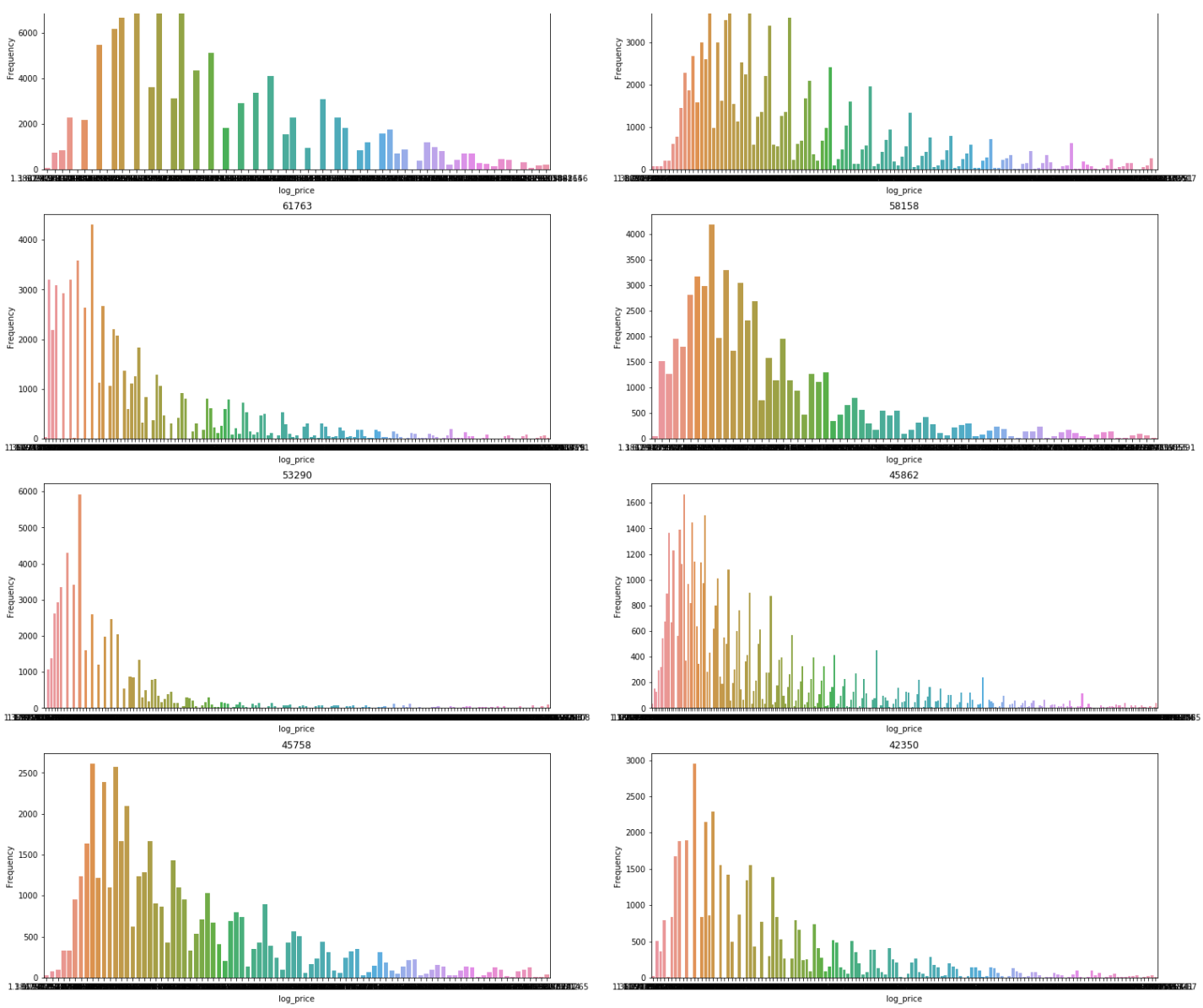
In [ ]:

```
chist = train.groupby(["subcat_1"], as_index = False).count().sort_values(by = "train_id
", ascending = False)[0:25]
```

In [ ]:

```
fig, ax = plt.subplots(5, 2, figsize = (25, 30))
k=10
for i in range(k):
    ohist = train.iloc[(nol(train[
        train["subcat_1"] == chist["subcat_1"].values[i]
    ].price).index).values].groupby(["log_price"], as_index = False).count()
    sns.barplot(x = ohist["log_price"], y = ohist["train_id"], ax = ax[int(i/2)][i%2])
    ax[int(i/2)][i%2].set_title(chist["category_name"].values[i])
    ax[int(i/2)][i%2].set_xlabel("log_price")
    ax[int(i/2)][i%2].set_ylabel("Frequency")
```





In [ ]:

```
```

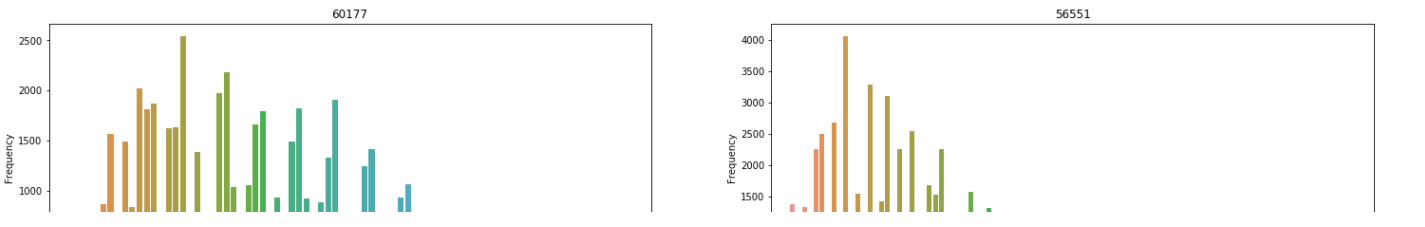
Price vs Frequency of Subcat_2

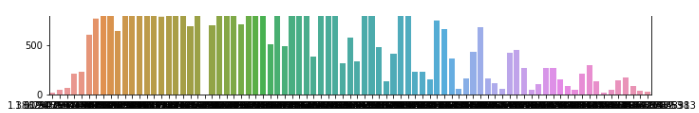
In [ ]:

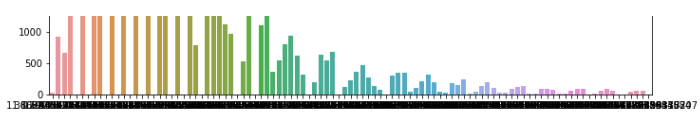
```
chist = train.groupby(["subcat_2"], as_index = False).count().sort_values(by = "train_id
", ascending = False)[0:25]
```

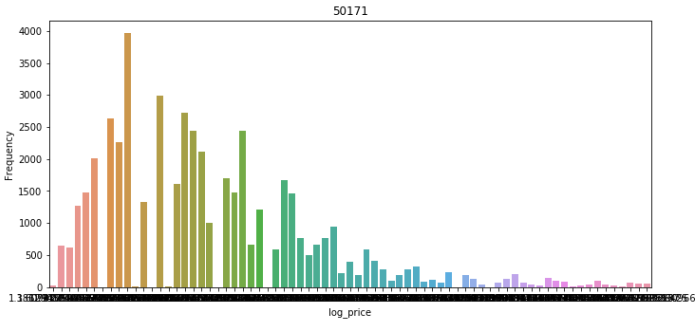
In [ ]:

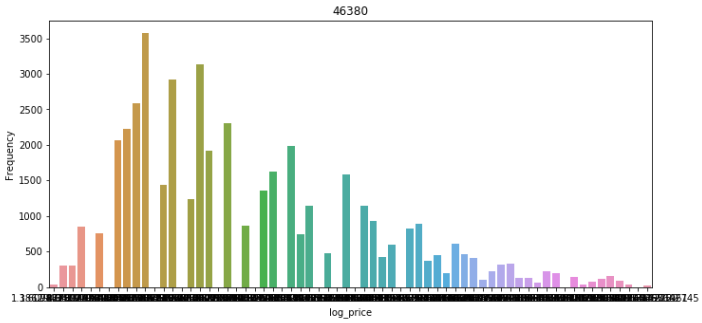
```
fig, ax = plt.subplots(5, 2, figsize = (25, 30))
k=10
for i in range(k):
    ohist = train.iloc[(nol(train[
        train["subcat_2"] == chist["subcat_2"].values[i]
    ].price).index).values].groupby(["log_price"], as_index = False).count()
    sns.barplot(x = ohist["log_price"], y = ohist["train_id"], ax = ax[int(i/2)][i%2])
    ax[int(i/2)][i%2].set_title(chist["category_name"].values[i])
    ax[int(i/2)][i%2].set_xlabel("log_price")
    ax[int(i/2)][i%2].set_ylabel("Frequency")
```

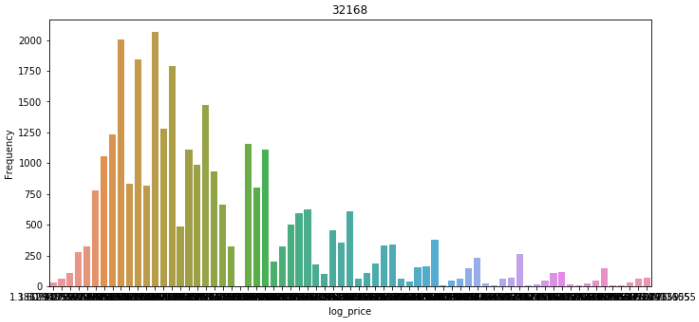


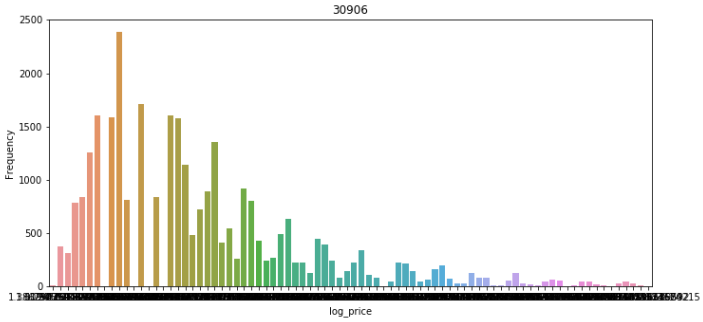

log_price

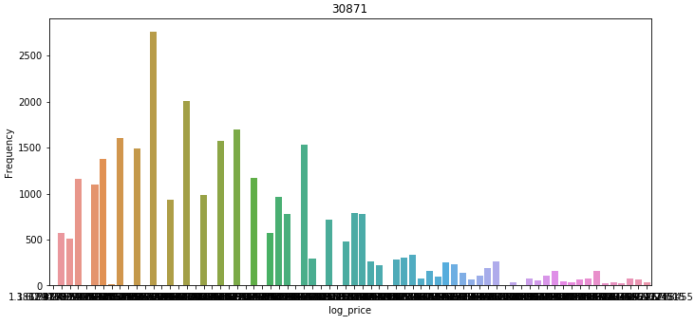

log_price

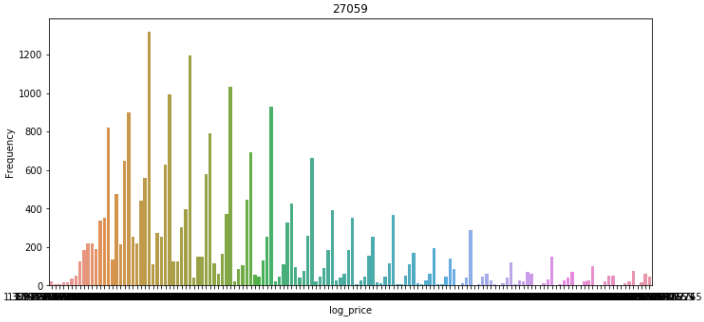

50171
Frequency
log_price

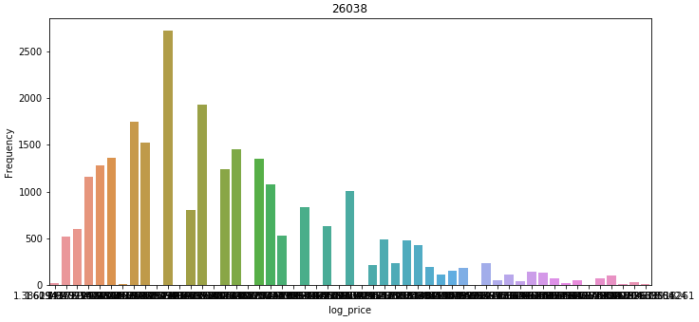

46380
Frequency
log_price

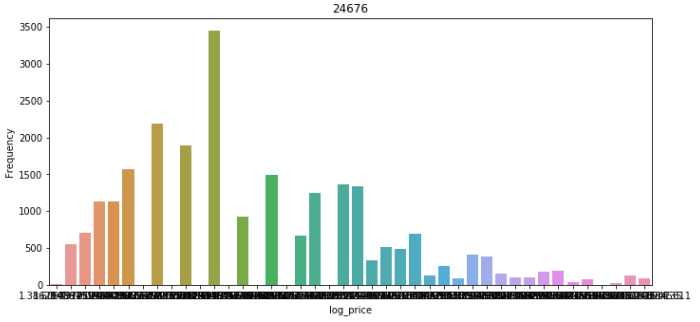

32168
Frequency
log_price


30906
Frequency
log_price


30871
Frequency
log_price


27059
Frequency
log_price


26038
Frequency
log_price


24676
Frequency
log_price

In [ ]:

In [ ]:

In [ ]:

In [ ]: