Predicting Future Sales

Project by Data Science - Group 22 (Batch 1)

- Amaan
- Jay
- Kishan

In this following project we are using following datasets:

- items_categories.csv
- items.csv
- sales_train.csv
- shops.csv
- test.csv

In this following project we would be performing following tasks:

- clearing of data
- taking out Monthly sales
- taking out yearly sales
- taking out most items sold per cathegory
- taking out items sold most in a particualar day of the month
- taking out busiest days of the shops
- taking out in which month people spend more money?
- taking out what of item category is famous among the customer?
- taking out which are the famous shops?
- predicting future sales

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import datetime
import plotly.express as px
import warnings
import lightgbm as lgbm
from lightgbm import LGBMRegressor
warnings.filterwarnings("ignore")
```

Provided data description

- sales_train.csv - the training set. Daily historical data from January 2013 to October 2015.

- test.csv - the test set. You need to forecast the sales for these shops and products for November 2015.
- sample_submission.csv - a sample submission file in the correct format.
- items.csv - supplemental information about the items/products.
- item_categories.csv - supplemental information about the items categories.
- shops.csv- supplemental information about the shops.

In [2]:
```
items = pd.read_csv('items.csv')
item_cat = pd.read_csv('item_categories.csv')
shops = pd.read_csv('shops.csv')
train = pd.read_csv('sales_train.csv')
test_items = pd.read_csv('test.csv')
```

In [3]:
```
items.head()
```

Out[3]:

	item_name	item_id	item_category_id
0	! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.) D	0	40
1	!ABBYY FineReader 12 Professional Edition Full...	1	76
2	***В ЛУЧАХ СЛАВЫ (UNV) D	2	40
3	***ГОЛУБАЯ ВОЛНА (Univ) D	3	40
4	***КОРОБКА (СТЕКЛО) D	4	40

In [4]:
```
item_cat.head()
```

Out[4]:

	item_category_name	item_category_id
0	PC - Гарнитуры/Наушники	0
1	Аксессуары - PS2	1
2	Аксессуары - PS3	2
3	Аксессуары - PS4	3
4	Аксессуары - PSP	4

In [5]:
```
shops.head()
```

Out[5]:

	shop_name	shop_id
0	!Якутск Орджоникидзе, 56 фран	0
1	!Якутск ТЦ "Центральный" фран	1
2	Адыгея ТЦ "Мега"	2
3	Балашиха ТРК "Октябрь-Киномир"	3
4	Волжский ТЦ "Волга Молл"	4

In [6]:
```
1 train.head()
```

Out[6]:

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	02.01.2013	0	59	22154	999.00	1.0
1	03.01.2013	0	25	2552	899.00	1.0
2	05.01.2013	0	25	2552	899.00	-1.0
3	06.01.2013	0	25	2554	1709.05	1.0
4	15.01.2013	0	25	2555	1099.00	1.0

In [7]:
```
1 test_items.head()
```

Out[7]:

	ID	shop_id	item_id
0	0	5	5037
1	1	5	5320
2	2	5	5233
3	3	5	5232
4	4	5	5268

In [8]:
```
1 train.shape
```

Out[8]: (2935849, 6)

In [9]:
```
1 train.isnull().sum()
```

Out[9]:
```
date              0
date_block_num    0
shop_id           0
item_id           0
item_price        0
item_cnt_day      0
dtype: int64
```

there are no nulls.

Monthly sales

In [10]:
```
1 monthly_sales=train.groupby(["date_block_num","shop_id","item_id"])[
2     "date","item_price","item_cnt_day"].agg({"date":["min",'max'],"item_price":"mean","item_cnt_day":"sum"})
```

In [11]:
```
monthly_sales
```

Out[11]:

			date		item_price	item_cnt_day
			min	max	mean	sum
date_block_num	shop_id	item_id				
0	0	32	03.01.2013	31.01.2013	221.0	6.0
		33	03.01.2013	28.01.2013	347.0	3.0
		35	31.01.2013	31.01.2013	247.0	1.0
		43	31.01.2013	31.01.2013	221.0	1.0
		51	13.01.2013	31.01.2013	128.5	2.0
...	...	...	...	...	...	...
33	59	22087	05.10.2015	23.10.2015	119.0	6.0
		22088	03.10.2015	27.10.2015	119.0	2.0
		22091	03.10.2015	03.10.2015	179.0	1.0
		22100	18.10.2015	18.10.2015	629.0	1.0
		22102	16.10.2015	16.10.2015	1250.0	1.0

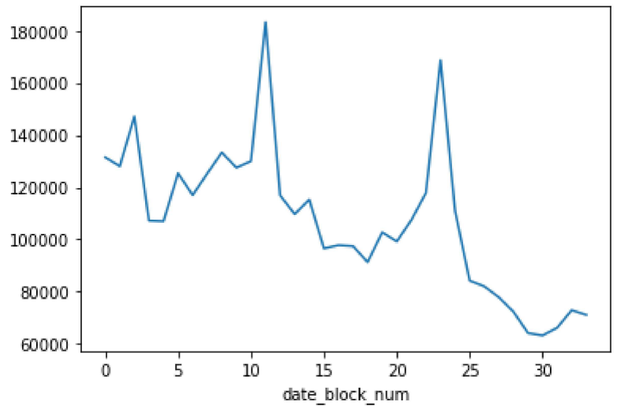
1609124 rows × 4 columns

In [12]:
```
monthly_sales.columns
```

Out[12]:
```
MultiIndex([(        'date',  'min'),
            (        'date',  'max'),
            (  'item_price', 'mean'),
            ('item_cnt_day',  'sum')],
           )
```

In [13]:
```
1 sales_by_month = train.groupby(['date_block_num'])['item_cnt_day'].sum()
2 sales_by_month.plot()
```

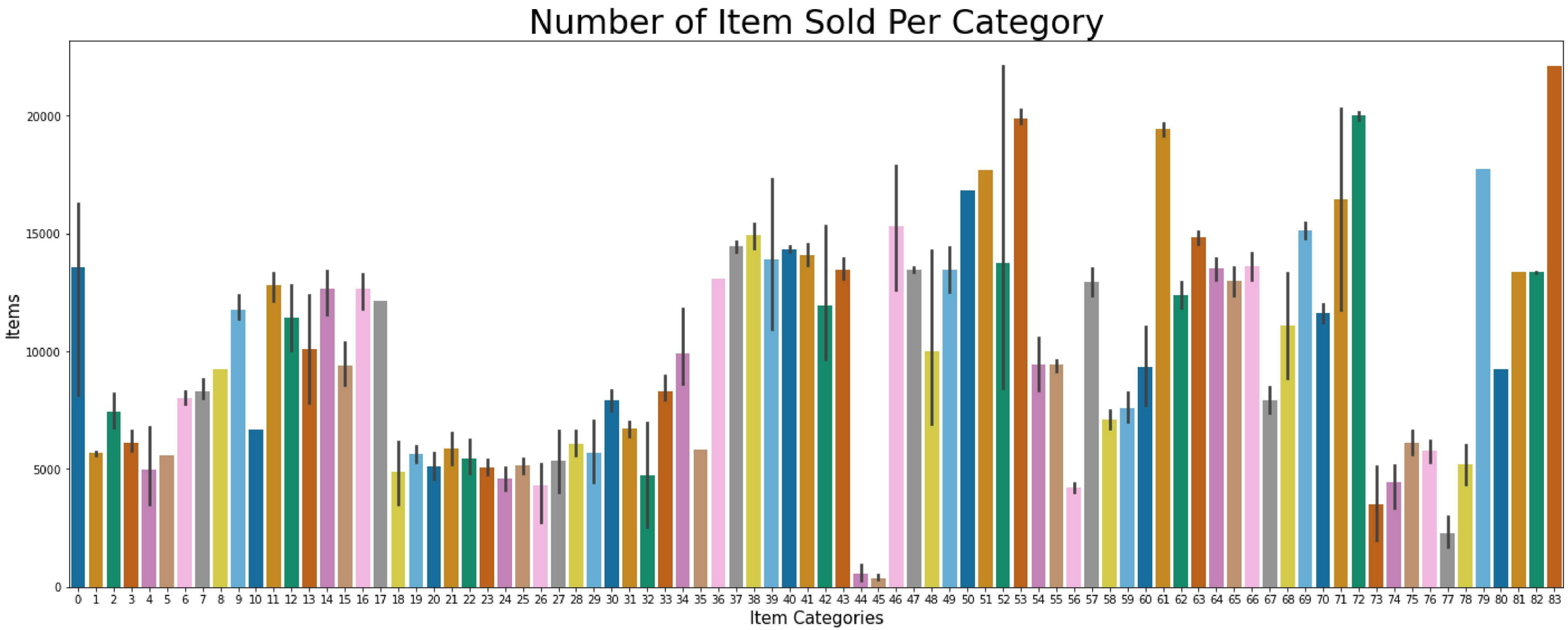
Out[13]: <AxesSubplot:xlabel='date_block_num'>



By looking at the plot above we can say that the sale is decreasing over months. However, some peaks are spotted during November.

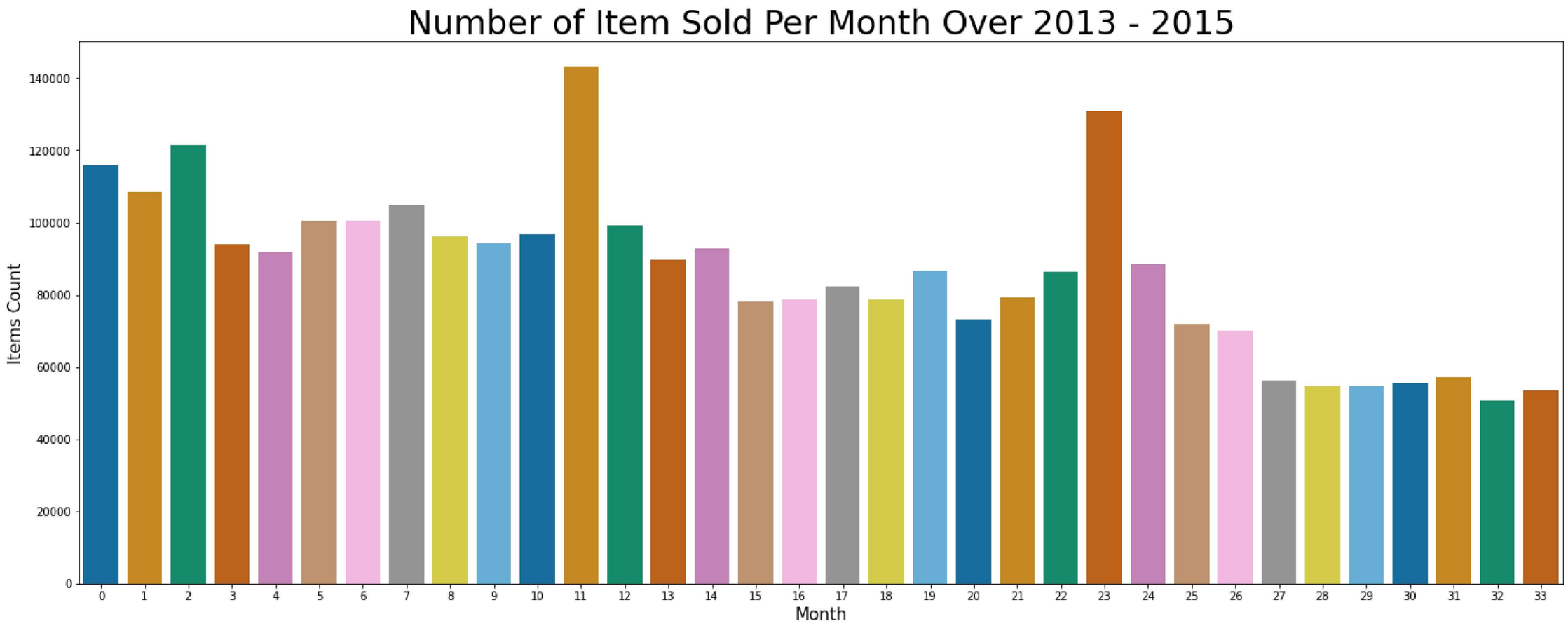
Checking how many items sold per category

In [14]:
```
plt.rcParams['figure.figsize'] = (24, 9)
sns.barplot(items['item_category_id'], items['item_id'], palette = 'colorblind')
plt.title('Number of Item Sold Per Category', fontsize = 30)
plt.xlabel('Item Categories', fontsize = 15)
plt.ylabel('Items', fontsize = 15)
plt.show()
```



Checking how many items sold per per month i.e. (jan 2013 ~ Oct 2015)

In [15]:
```
plt.rcParams['figure.figsize'] = (24, 9)
sns.countplot(train['date_block_num'], palette = 'colorblind')
plt.title('Number of Item Sold Per Month Over 2013 - 2015', fontsize = 30)
plt.xlabel('Month', fontsize = 15)
plt.ylabel('Items Count', fontsize = 15)
plt.show()
```



checking the number of unique snop names and item category names

In [16]:
```
# item_cat['item_category_name'].count()
print(item_cat['item_category_name'].nunique())
print(shops['shop_name'].nunique())
```

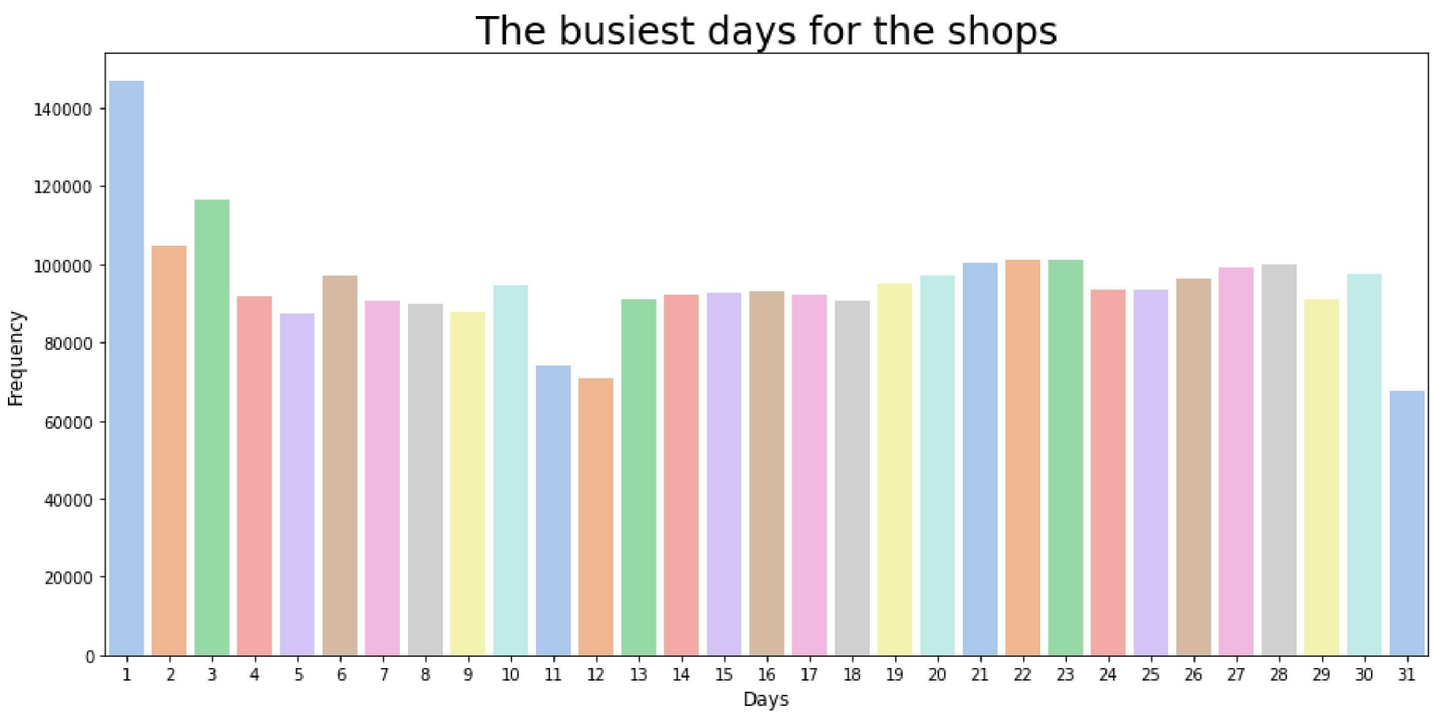
```
84
60
```

Busiest days for the shop

In [17]:
```
train['date'] = pd.to_datetime(train['date'], errors='coerce')
```

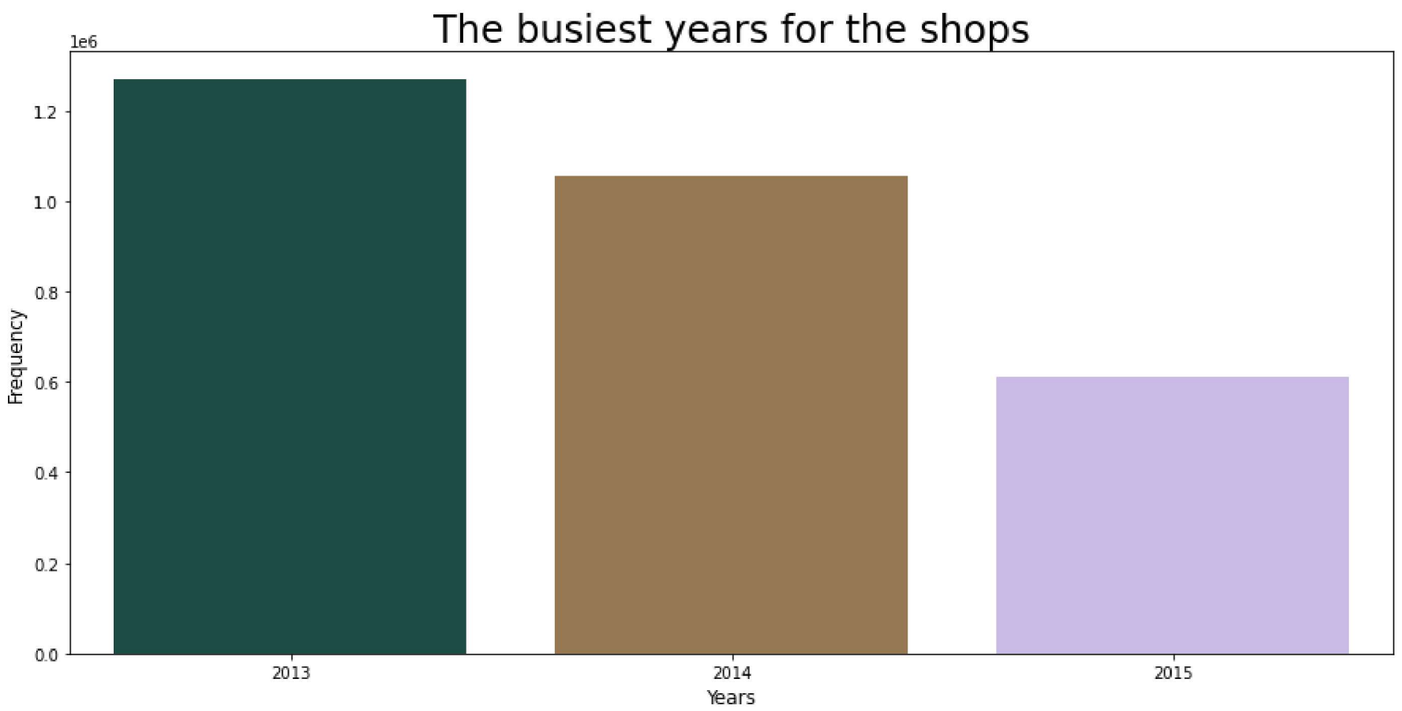
In [18]:
```
days = []
months = []
years = []

for day in train['date']:
    days.append(day.day)
for month in train['date']:
    months.append(month.month)
for year in train['date']:
    years.append(year.year)
```

In [19]:
```
plt.rcParams['figure.figsize'] = (15, 7)
sns.countplot(days, palette= 'pastel')
plt.title('The busiest days for the shops', fontsize = 24)
plt.xlabel('Days', fontsize = 12)
plt.ylabel('Frequency', fontsize = 12)

plt.show()
```



Busiest Months and years for the shops

In [20]:
```
# busy month
plt.rcParams['figure.figsize'] = (15, 7)
sns.countplot(months, palette= 'rocket')
plt.title('The busiest months for the shops', fontsize = 24)
plt.xlabel('Months', fontsize = 12)
plt.ylabel('Frequency', fontsize = 12)

plt.show()

# busy year
plt.rcParams['figure.figsize'] = (15, 7)
sns.countplot(years, palette= 'cubehelix')
plt.title('The busiest years for the shops', fontsize = 24)
plt.xlabel('Years', fontsize = 12)
plt.ylabel('Frequency', fontsize = 12)

plt.show()
```



In [21]:
```
train['day'] = days
train['month'] = months
train['year'] = years
```

In [22]:
```
train
```

Out[22]:

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day	day	month	year
0	2013-02-01	0	59	22154	999.00	1.0	1	2	2013
1	2013-03-01	0	25	2552	899.00	1.0	1	3	2013
2	2013-05-01	0	25	2552	899.00	-1.0	1	5	2013
3	2013-06-01	0	25	2554	1709.05	1.0	1	6	2013
4	2013-01-15	0	25	2555	1099.00	1.0	15	1	2013
...	...	...	...	...	...	...	...	...	...
2935844	2015-10-10	33	25	7409	299.00	1.0	10	10	2015
2935845	2015-09-10	33	25	7460	299.00	1.0	10	9	2015
2935846	2015-10-14	33	25	7459	349.00	1.0	14	10	2015
2935847	2015-10-22	33	25	7440	299.00	1.0	22	10	2015
2935848	2015-03-10	33	25	7460	299.00	1.0	10	3	2015

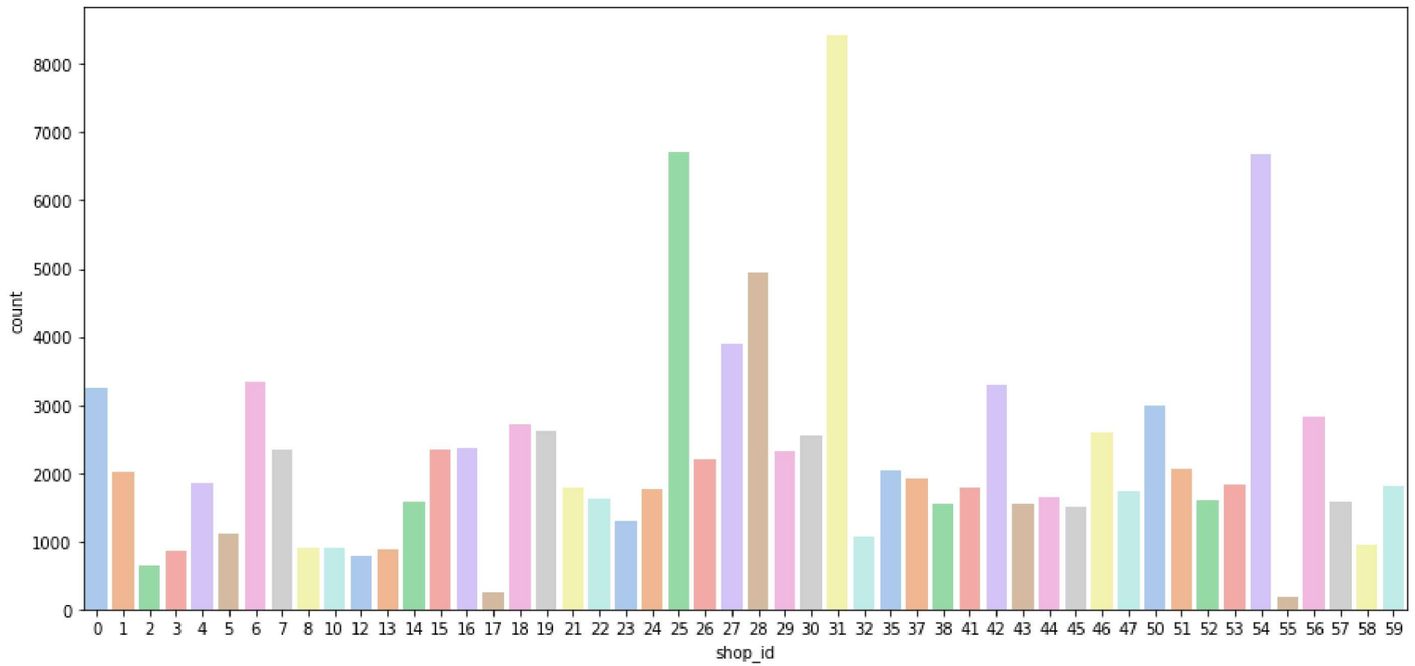
2935849 rows × 9 columns

Note that the list of shops and products slightly changes every month.

we can see that in Feb of 2013 `shop_id` `31` has the highest number of sales

In [23]:
```
1 sns.countplot(train[(train.month == 2) & (train.year == 2013)]['shop_id'], palette='pastel')
```

Out[23]: <AxesSubplot:xlabel='shop_id', ylabel='count'>



Outliers

In [24]:
```
1 train.describe()
```

Out[24]:

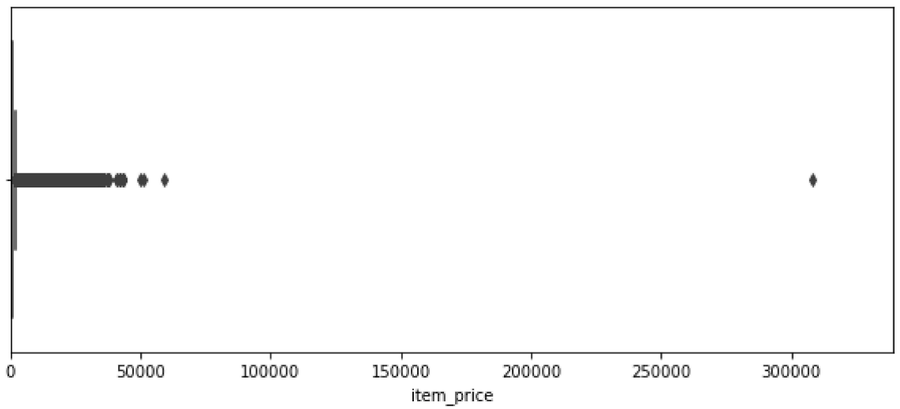
	date_block_num	shop_id	item_id	item_price	item_cnt_day	day	month	year
count	2.935849e+06	2.935849e+06	2.935849e+06	2.935849e+06	2.935849e+06	2.935849e+06	2.935849e+06	2.935849e+06
mean	1.456991e+01	3.300173e+01	1.019723e+04	8.908532e+02	1.242641e+00	1.566783e+01	6.432552e+00	2.013777e+03
std	9.422988e+00	1.622697e+01	6.324297e+03	1.729800e+03	2.618834e+00	9.128682e+00	3.504695e+00	7.684790e-01
min	0.000000e+00	0.000000e+00	0.000000e+00	-1.000000e+00	-2.200000e+01	1.000000e+00	1.000000e+00	2.013000e+03
25%	7.000000e+00	2.200000e+01	4.476000e+03	2.490000e+02	1.000000e+00	7.000000e+00	3.000000e+00	2.013000e+03
50%	1.400000e+01	3.100000e+01	9.343000e+03	3.990000e+02	1.000000e+00	1.600000e+01	6.000000e+00	2.014000e+03
75%	2.300000e+01	4.700000e+01	1.568400e+04	9.990000e+02	1.000000e+00	2.400000e+01	9.000000e+00	2.014000e+03
max	3.300000e+01	5.900000e+01	2.216900e+04	3.079800e+05	2.169000e+03	3.100000e+01	1.200000e+01	2.015000e+03

from the description above, when we look at the max and min values we can see that there is an outlier for item_price and item_cnt_day

Below we plot the Outliners

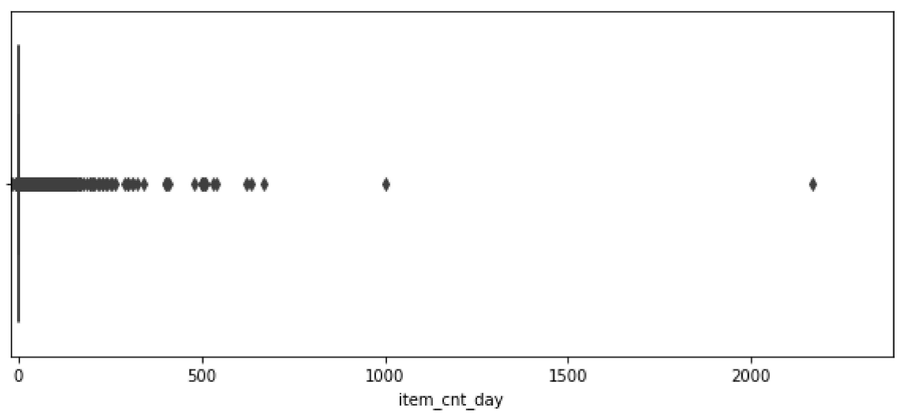
In [25]:
```
plt.figure(figsize=(10,4))
plt.xlim(train.item_price.min(), train.item_price.max()*1.1)
sns.boxplot(x=train.item_price)
```

Out[25]: <AxesSubplot:xlabel='item_price'>



In [26]:
```
plt.figure(figsize=(10,4))
plt.xlim(train.item_cnt_day.min(), train.item_cnt_day.max()*1.1)
sns.boxplot(x=train.item_cnt_day)
```

Out[26]: <AxesSubplot:xlabel='item_cnt_day'>



By judging from the above outlier diagram above we see a price point further than the other points. So we can get rid of that point.

Also for item_cnt_day there is a point further than other point we will get rid of that point too.

The demonstration has been shown below:

In [27]:
```
train = train[(train["item_price"] > 0) & (train["item_price"] < 50000)]
train = train[(train["item_cnt_day"] > 0) & (train["item_cnt_day"] < 1000)]
```

In [28]:
```
train.shape
```

Out[28]: (2928487, 9)

Also, from the diagram we can see that there is a price point that is less than zero. We will fill that price with median value.

In [29]:
```
train[train['item_price'] < 0]
```

Out[29]:

date	date_block_num	shop_id	item_id	item_price	item_cnt_day	day	month	year

In [30]:
```
median = train[(train.shop_id==32)&(train.item_id==2973)&(train.date_block_num==4)&(train.item_price>0)].item_price.median()
median
```

Out[30]: 1874.0

After assigning the median value we can no longer find any record with negative pricing.

In [31]:
```
train["item_price"] = train["item_price"].map(lambda x: median if x<0 else x)
```

No `item_price` less than 0 remaining

In [32]:
```
train[train['item_price'] < 0]
```

Out[32]:

date	date_block_num	shop_id	item_id	item_price	item_cnt_day	day	month	year

We can also see from the (2nd outlier) `item_cnt_day` diagram that there are some negative values.

In [33]:
```
train[train['item_cnt_day'] < 0]
```

Out[33]:

date	date_block_num	shop_id	item_id	item_price	item_cnt_day	day	month	year

no < 0 `item_cnt_day` value remaining

Merging Dataset

In [34]:
```
df1=pd.merge(items,item_cat,on='item_category_id',how='inner')
df2=pd.merge(shops,train,on="shop_id",how='inner' )
df=pd.merge(df1,df2,on="item_id",how="inner")
```

In [35]:
```
df
```

Out[35]:

	item_name	item_id	item_category_id	item_category_name	shop_name	shop_id	date	date_block_num	item_price	item_cnt_day	day	month	year
0	! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.) D	0	40	Кино - DVD	Химки ТЦ "Мега"	54	2014-01-09	20	58.0	1.0	9	1	2014
1	***В ЛУЧАХ СЛАВЫ (UNV) D	2	40	Кино - DVD	Химки ТЦ "Мега"	54	2014-08-24	19	58.0	1.0	24	8	2014
2	***В ЛУЧАХ СЛАВЫ (UNV) D	2	40	Кино - DVD	Химки ТЦ "Мега"	54	2014-12-11	22	58.0	1.0	11	12	2014
3	***ГОЛУБАЯ ВОЛНА (Univ) D	3	40	Кино - DVD	Химки ТЦ "Мега"	54	2014-05-07	18	100.0	1.0	7	5	2014
4	***ГОЛУБАЯ ВОЛНА (Univ) D	3	40	Кино - DVD	Химки ТЦ "Мега"	54	2014-08-26	19	58.0	1.0	26	8	2014
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2928482	Элемент питания GP 24AU (LR03) - Блистер 2 шт ...	22097	83	Элементы питания	Коломна ТЦ "Рио"	16	2013-02-24	1	60.0	1.0	24	2	2013
2928483	Элемент питания КОСМОС LR03 2*BL	22098	83	Элементы питания	Жуковский ул. Чкалова 39м?	10	2013-11-12	11	7.0	1.0	12	11	2013
2928484	Элемент питания СТАРТ ZT 15A (LR6) - Блистер 2 шт	22099	83	Элементы питания	Адыгея ТЦ "Мега"	2	2013-07-19	6	40.0	1.0	19	7	2013
2928485	Элемент питания СТАРТ ZT 15A (LR6) - Блистер 2 шт	22099	83	Элементы питания	Жуковский ул. Чкалова 39м?	10	2013-11-12	11	23.0	1.0	12	11	2013
2928486	Элемент питания СТАРТ ZT 15A (LR6) - Блистер 2 шт	22099	83	Элементы питания	Коломна ТЦ "Рио"	16	2013-11-20	10	40.0	1.0	20	11	2013

2928487 rows × 13 columns

now we will keep the data which is also present in test data

In [36]:
```
df = df[df['shop_id'].isin(test_items['shop_id'].unique())]
df = df[df['item_id'].isin(test_items['item_id'].unique())]

```

In [37]:
```
'''l1=[i for i in df.columns]
l2=[df[i].nunique() for i in df.columns]
l1 = pd.DataFrame(l1)
l2 = pd.DataFrame(l2)
df2 = pd.concat([l1,l2],axis=1)'''
```

Out[37]:
```
'l1=[i for i in df.columns]\nl2=[df[i].nunique() for i in df.columns]\nl1 = pd.DataFrame(l1)\nl2 = pd.DataFrame(l2)\ndf2 = pd.concat([l1,l2],axis=1)'
```

DF2

In [38]:
```
def uni(df):
    l1=[i for i in df.columns]
    l2=[df[i].nunique() for i in df.columns]
    l1 = pd.DataFrame(l1)
    l2 = pd.DataFrame(l2)
    df2 = pd.concat([l1,l2],axis=1)
    df2.columns = ['Names','Unique_values']
    plt.figure(figsize=(18,7))
    g = sns.barplot(x = 'Names',y = 'Unique_values',data = df2)
    for index, row in df2.iterrows():
        g.text(row.name,row.Unique_values,round(row.Unique_values,2),color='black', ha="center")
```

now we will featur engineering date column and break it into day and month , so we can extract more information with respect to time

In [39]:
```
df['date'] = pd.to_datetime(df['date'])
df['day'] = df['date'].dt.day
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year
```

In [40]:
```
df
```

Out[40]:

	item_name	item_id	item_category_id	item_category_name	shop_name	shop_id	date	date_block_num	item_price	item_cnt_day	day	month	year
44	007: КООРДИНАТЫ «СКАЙФОЛЛ»	30	40	Кино - DVD	Адыгея ТЦ "Мега"	2	2013-01-03	2	359.0	1.0	3	1	2013
45	007: КООРДИНАТЫ «СКАЙФОЛЛ»	30	40	Кино - DVD	Адыгея ТЦ "Мега"	2	2013-06-16	5	399.0	1.0	16	6	2013
46	007: КООРДИНАТЫ «СКАЙФОЛЛ»	30	40	Кино - DVD	Адыгея ТЦ "Мега"	2	2014-04-20	15	169.0	1.0	20	4	2014
47	007: КООРДИНАТЫ «СКАЙФОЛЛ»	30	40	Кино - DVD	Адыгея ТЦ "Мега"	2	2014-08-05	16	169.0	1.0	5	8	2014
48	007: КООРДИНАТЫ «СКАЙФОЛЛ»	30	40	Кино - DVD	Балашиха ТРК "Октябрь-Киномир"	3	2013-02-23	1	399.0	1.0	23	2	2013
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2928467	Элемент питания DURACELL TURBO LR6 2*BL	22092	83	Элементы питания	Ярославль ТЦ "Альтаир"	59	2015-06-01	24	139.0	1.0	1	6	2015
2928468	Элемент питания DURACELL TURBO LR6 2*BL	22092	83	Элементы питания	Ярославль ТЦ "Альтаир"	59	2015-02-01	24	139.0	1.0	1	2	2015
2928469	Элемент питания DURACELL TURBO LR6 2*BL	22092	83	Элементы питания	Ярославль ТЦ "Альтаир"	59	2015-03-21	26	179.0	1.0	21	3	2015
2928470	Элемент питания DURACELL TURBO LR6 2*BL	22092	83	Элементы питания	Ярославль ТЦ "Альтаир"	59	2015-03-24	26	179.0	2.0	24	3	2015
2928471	Элемент питания DURACELL TURBO LR6 2*BL	22092	83	Элементы питания	Ярославль ТЦ "Альтаир"	59	2015-08-18	31	179.0	1.0	18	8	2015

1221495 rows × 13 columns

In [41]:
```
l = df['date'].dt.year.unique()
print(f'Years are {l}')
```

```
Years are [2013 2014 2015]
```

We can see that there are 3 diffrent years - '2013','2014','2015'

So now we will explore each year seperately

In [42]:
```
#2013
df_2013 = (df[df['date'].dt.year == 2013]).drop('date',axis = 1)
#2014
df_2014 = (df[df['date'].dt.year == 2014].reset_index()).drop('date',axis = 1)

#2015
df_2015 = (df[df['date'].dt.year == 2015].reset_index()).drop('date',axis = 1)
```

To analyse for each particular year

- Which is the most expensive product of in every month seprately?
- In which month people spend more money?
- What of item category is famous among the customer with help of pie chart?
- Which are the famous shops?

2013

In [43]:
```
#most expensive product of in every month seprately

# this function will give top expensive products of every months in form of DataFrame

def func1(df):
    new_df = pd.DataFrame()
    for i in range(1,13):
        temp = df[df['month']==i]
        temp.sort_values(by='item_price',inplace = True,ascending = False)
        new_df = pd.concat([temp[:1],new_df])
    return new_df.sort_values(by = 'month')
```

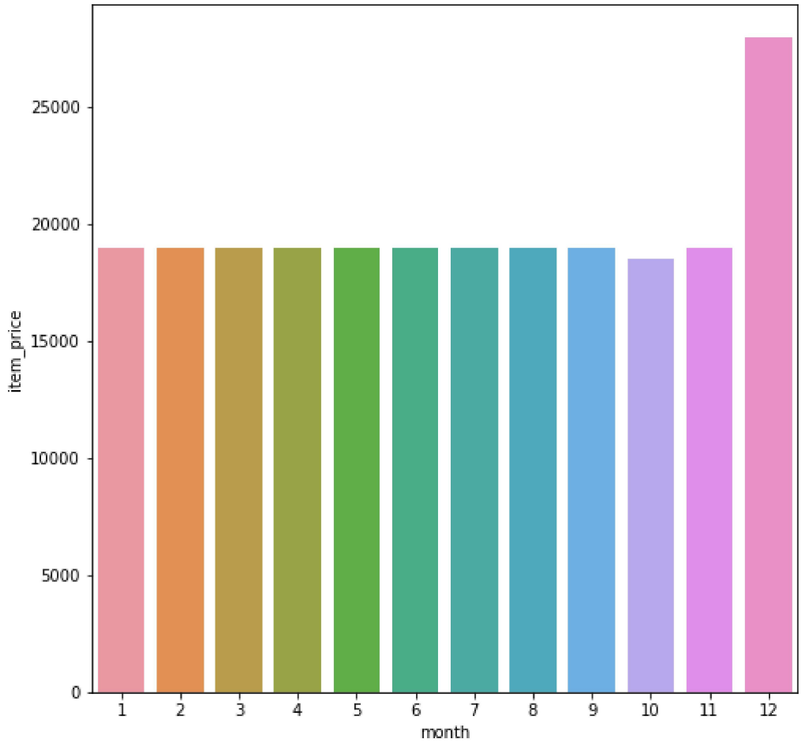
In [44]:
```
temp1 = func1(df_2013)
temp1
```

Out[44]:

	item_name	item_id	item_category_id	item_category_name	shop_name	shop_id	date_block_num	item_price	item_cnt_day	day	month	year
2793349	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Балашиха ТРК "Октябрь-Киномир"	3	11	18990.0	1.0	12	1	2013
2795822	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Москва ТЦ "Семеновский"	31	11	18990.0	2.0	12	2	2013
2798125	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Якутск ТЦ "Центральный"	58	11	18990.0	2.0	12	3	2013
2796457	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	РостовНаДону ТЦ "Мега"	41	11	18990.0	1.0	12	4	2013
2798257	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Ярославль ТЦ "Альтаир"	59	11	18990.0	2.0	12	5	2013
2797624	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Уфа ТЦ "Семья" 2	53	11	18990.0	2.0	12	6	2013
2797625	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Уфа ТЦ "Семья" 2	53	11	18990.0	3.0	12	7	2013
2798128	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Якутск ТЦ "Центральный"	58	11	18990.0	1.0	12	8	2013
2797116	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Сургут ТРЦ "Сити Молл"	47	11	18990.0	1.0	12	9	2013
2798130	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Якутск ТЦ "Центральный"	58	11	18490.0	2.0	12	10	2013
2797626	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Уфа ТЦ "Семья" 2	53	11	18990.0	1.0	12	11	2013
2797003	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	6675	12	Игровые консоли - PS4	Сергиев Посад ТЦ "7Я"	46	11	27990.0	1.0	23	12	2013

In [45]:
```
# now we will analyse in which month people spend more money

# this function will give two list x,y
def func2(df):
    temp2 = func1(df)
    plt.rcParams['figure.figsize'] = [8, 8]
    sns.barplot(data=temp2, x="month", y="item_price")
    plt.show()
```

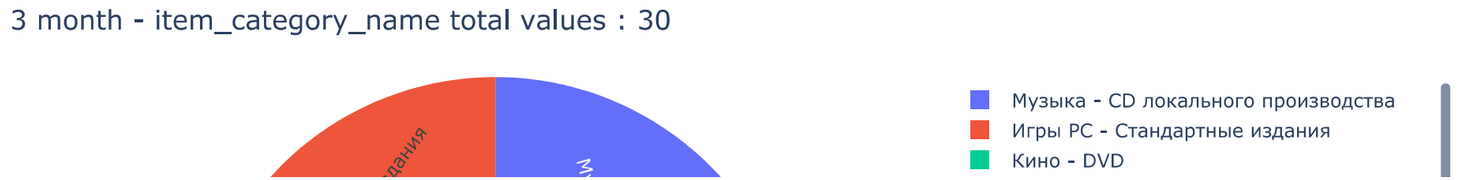
In [46]:
```
1 func2(df_2013)
```



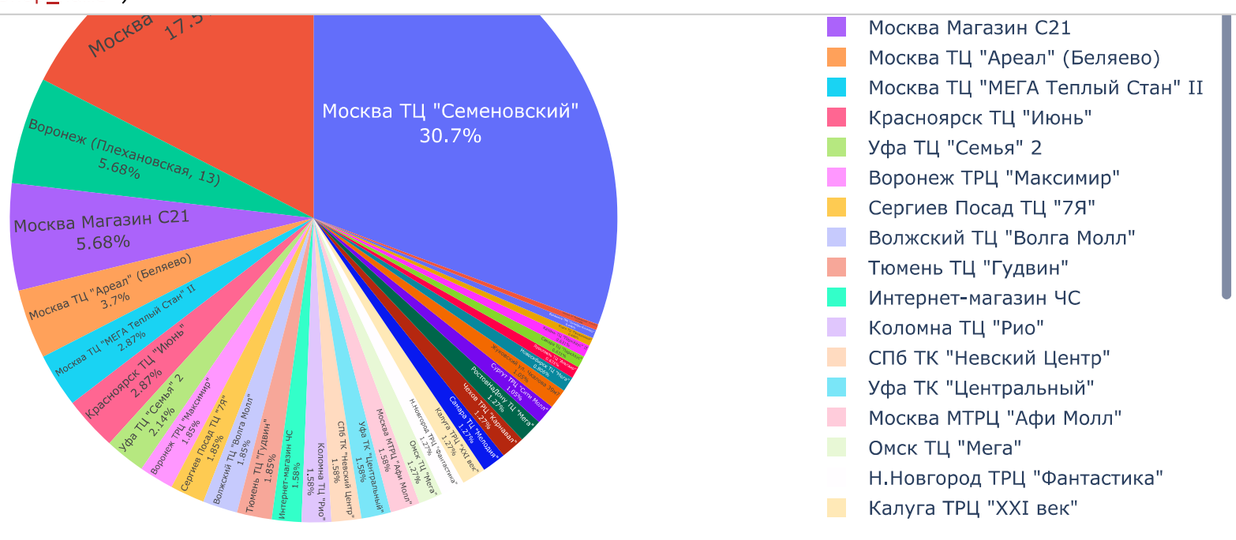
In [47]:

```
# now we will see, what kind of item category is famous among the customer with help of pie chart we will visualise


# this function give customised dataframe for a particular month sorted by feat
def func3(df,m,y,feat):
    from calendar import monthrange
    df1 = df[df['month']==m]
    d = list(monthrange(y, m))[1]
    for i in  range(1,d+1):
        df2 = df1[df['day']==i]
        df2.sort_values(by=feat,inplace = True,ascending = False)
    return df2

# this function creates a dataframe of percent of count in context with feat1
def func4(df,feat1):
    lis = list(df[feat1])
    l = len(lis)
    values = []
    for i in lis:
        c = lis.count(i)
        values.append(round((c/l)*100,1))
    names  = lis
    df2 = pd.DataFrame(list(zip(names,values)),columns =['names', 'values'])
    return df2


# this function plots piechart
def func5(df2,m,feat2):
    l = df2['names'].nunique()
    fig = px.pie(df2, values='values', names='names')
    fig.update_traces(textposition='inside', textinfo='percent+label')
    fig.update_layout(title = dict(text = f'{m} month - {feat2} total values : {l}'),)
    fig.show()
```

In [48]:
```
for i in range(1,13):
    temp1 = func3(df_2013,1,2013,'item_price')
    temp2 = func4(temp1,'item_category_name')
    func5(temp2,i,'item_category_name')
```

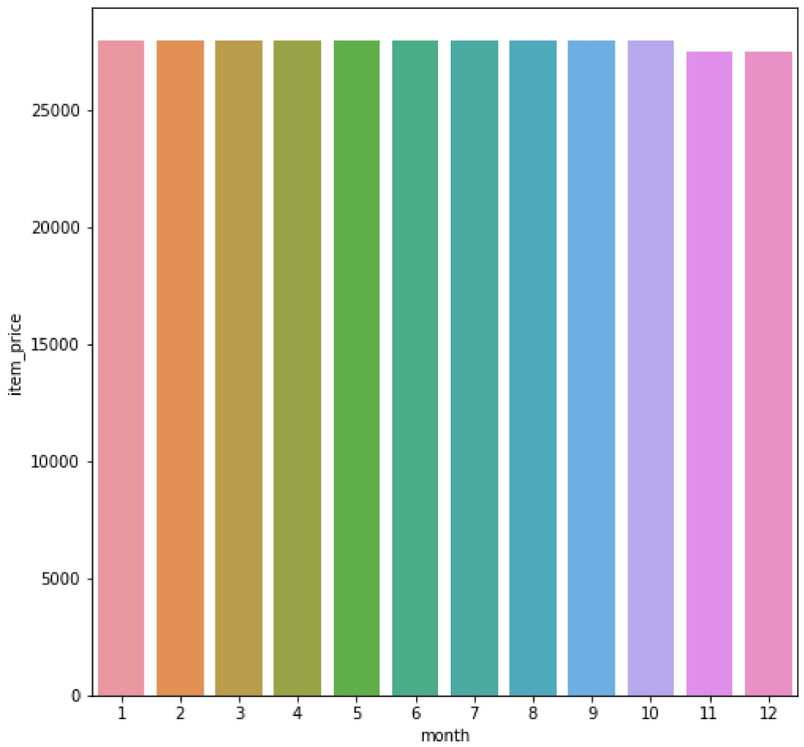


In [49]:
```
# now we will visualise favourite shops of customers with help of chart
# shop id and shop_name are basically same so we will use shop_name
# and during time of model making we will drop one of them

for i in range(1,13):
    temp1 = func3(df_2013,1,2013,'item_price')
    temp2 = func4(temp1,'shop_name')
    func5(temp2,i,'shop_name')
```

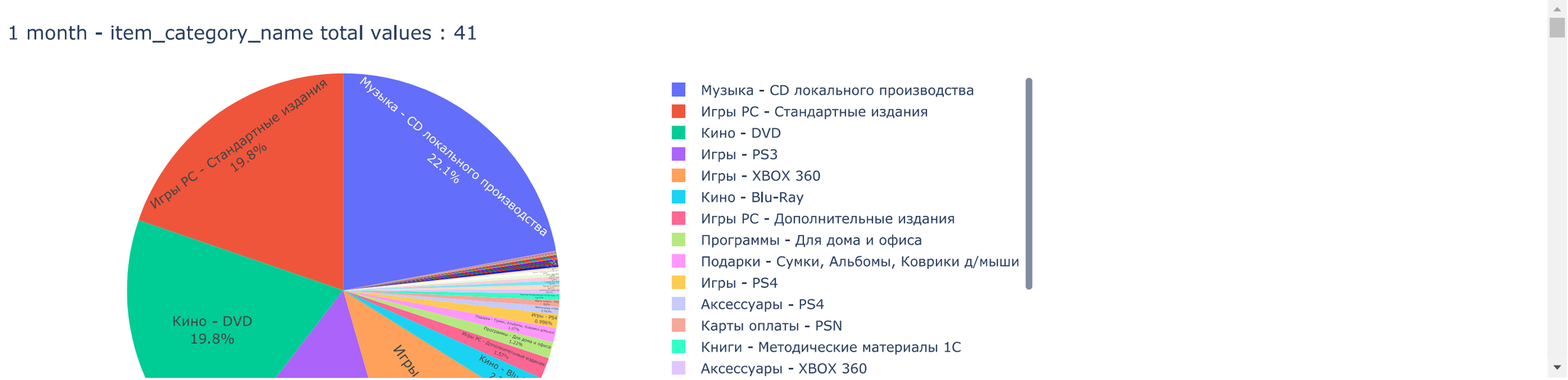


2014

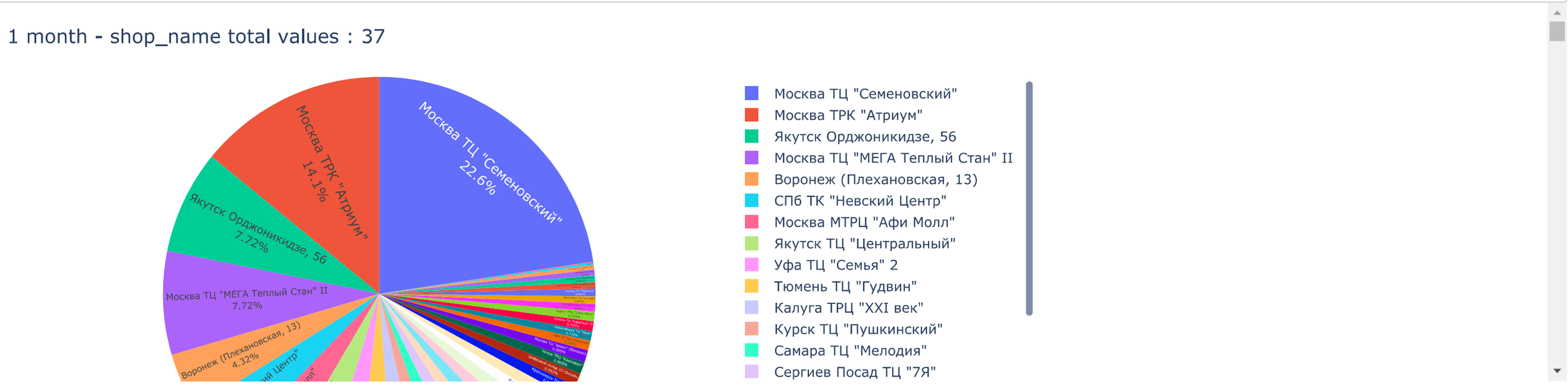
In [50]:
```
temp1 = func1(df_2014)
func2(temp1)
```



In [51]:
```
for i in range(1,13):
    temp1 = func3(df_2014,1,2014,'item_price')
    temp2 = func4(temp1,'item_category_name')
    func5(temp2,i,'item_category_name')
```

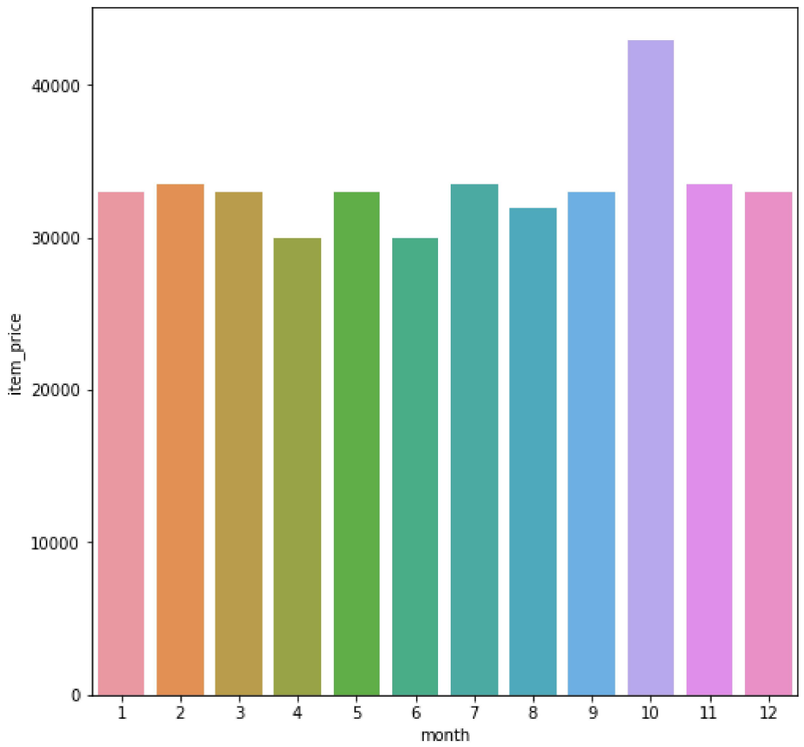


In [52]:
```
for i in range(1,13):
    temp1 = func3(df_2014,1,2014,'item_price')
    temp2 = func4(temp1,'shop_name')
    func5(temp2,i,'shop_name')
```

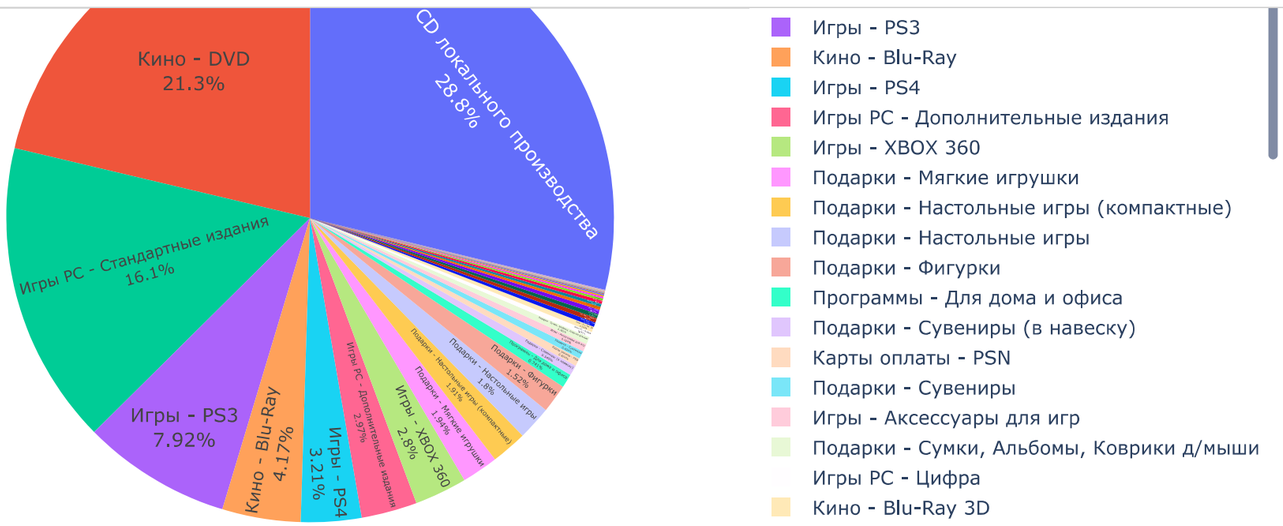


2015

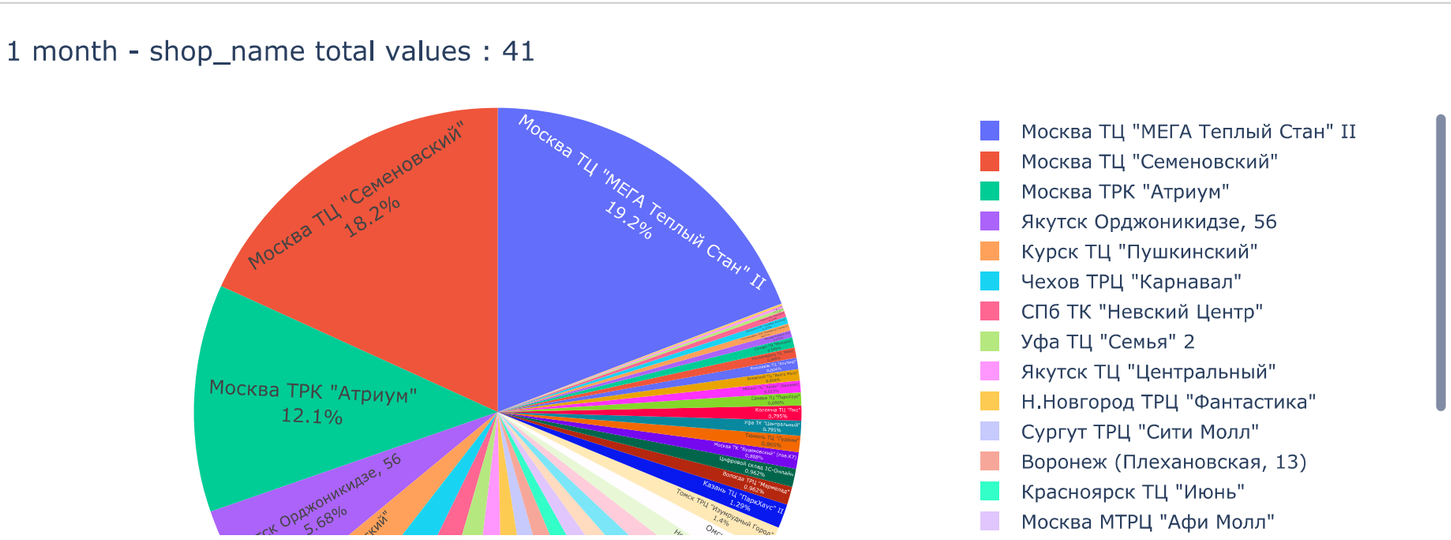
In [53]:
```
temp1 = func1(df_2015)
func2(temp1)
```



In [54]:
```
for i in range(1,13):
    temp1 = func3(df_2015,1,2015,'item_price')
    temp2 = func4(temp1,'item_category_name')
    func5(temp2,i,'item_category_name')
```



In [55]:
```
for i in range(1,13):
    temp1 = func3(df_2015,1,2015,'item_price')
    temp2 = func4(temp1,'shop_name')
    func5(temp2,i,'shop_name')
```

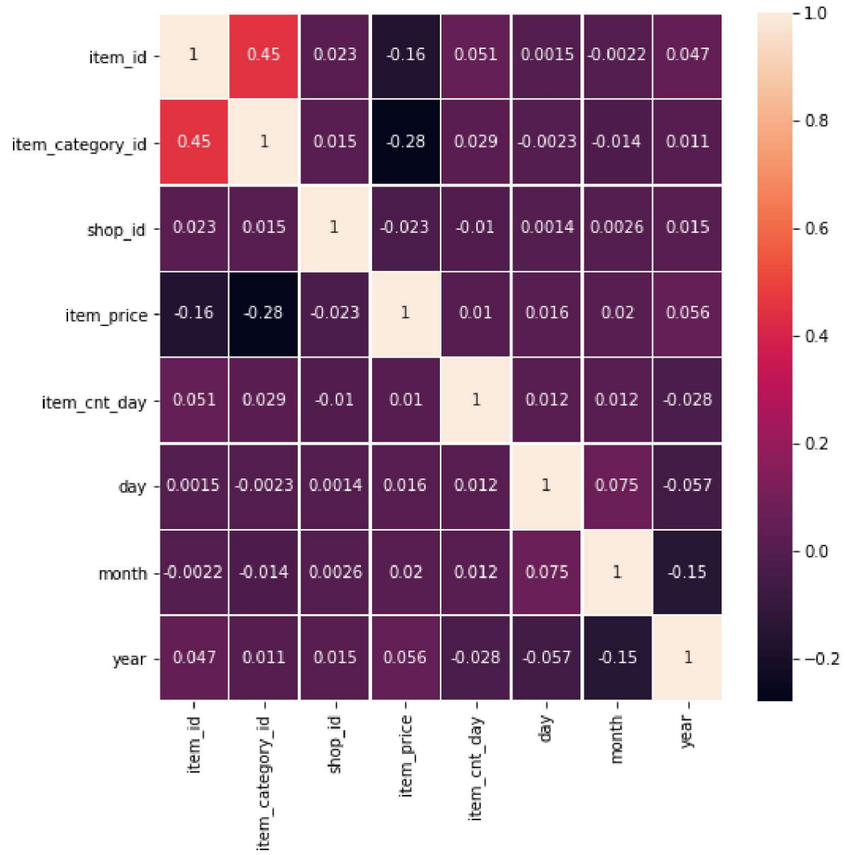


In [56]:
```
# now i guess its enough for basic visualisation
# now we will do some feature engineering
```

In [57]:
```
# droping irrelevant features
df = df.drop(['date', 'date_block_num','shop_name', 'item_name','item_category_name'],axis = 1)
```

In [58]:
```
sns.heatmap(df.corr(),linewidths=0.5,linecolor='white',annot=True)
```

Out[58]: <AxesSubplot:>



In [59]:
```
x = df[['shop_id', 'item_id', 'item_cnt_day', 'item_category_id','day', 'month','year']]
y = df['item_price']
```

In [60]:
```
# now we will split the data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,shuffle = False,test_size = 0.3)
```

In [61]:
```
# now we will import linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

Out[61]: LinearRegression()

In [62]:
```
from sklearn.metrics import mean_squared_error
print('Train set mse:', mean_squared_error(y_train, model.predict(x_train)))
print('Test set mse:', mean_squared_error(y_test, model.predict(x_test)))
print('Test set score:', model.score(x_train,y_train))
```

```
Train set mse: 486062.23301737197
Test set mse: 9706062.489780499
Test set score: 0.1189574097250945
```

Data Processing

In [63]:
```
train.head(2)
```

Out[63]:

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day	day	month	year
0	2013-02-01	0	59	22154	999.0	1.0	1	2	2013
1	2013-03-01	0	25	2552	899.0	1.0	1	3	2013

In [64]:
```
print("total unique items: ", items['item_id'].nunique())
print("total unique items in train dataset: ", train['item_id'].nunique())
print("total unique items in test dataset: ", test_items['item_id'].nunique())

print("total unique shops: ", shops['shop_id'].nunique())
print("total unique shops in train dataset: ", train['shop_id'].nunique())
print("total unique shops in test dataset: ", test_items['shop_id'].nunique())
```

```
total unique items:  22170
total unique items in train dataset:  21802
total unique items in test dataset:  5100
total unique shops:  60
total unique shops in train dataset:  60
total unique shops in test dataset:  42
```

we can see that item numbers in test and train sets are not equal. So making prediction for the missing items is going to be difficult.

Lets find out which `item_ids` are in test_set but not in train_set

In [65]:
```
test_item_list = [x for x in (np.unique(test_items['item_id']))]
train_item_list = [x for x in (np.unique(train['item_id']))]

missing_item_ids_ = [element for element in test_item_list if element not in train_item_list]
len(missing_item_ids_)
```

Out[65]: 363

363 items are not found in `train_dataset` so predicting sales for these items is not easy since we do not have the prices for these items.

Processing shop data

Lets Look at all the Shops now. Every shop_name is designed like this: shop_name = city + kind of shop.

We attempt to extract the city feature out of the shop_name to add more diversity to our dataset.

The first two row shows that a city name is starting with '!', so we will get rid of the '!'.

Index 46 gives us a shop_name as **Сергиев Посад ТЦ "7Я"**

IDK Russian but analyzing different kernels it seem like the city name is actually **СергиевПосад** not **Сергиев Посад**, there is an extra " ". So we will get rid of that too.

In [66]:
```
1 shops
```

Out[66]:

	shop_name	shop_id
0	!Якутск Орджоникидзе, 56 фран	0
1	!Якутск ТЦ "Центральный" фран	1
2	Адыгея ТЦ "Мега"	2
3	Балашиха ТРК "Октябрь-Киномир"	3
4	Волжский ТЦ "Волга Молл"	4
5	Вологда ТРЦ "Мармелад"	5
6	Воронеж (Плехановская, 13)	6
7	Воронеж ТРЦ "Максимир"	7
8	Воронеж ТРЦ Сити-Парк "Град"	8
9	Выездная Торговля	9
10	Жуковский ул. Чкалова 39м?	10
11	Жуковский ул. Чкалова 39м²	11
12	Интернет-магазин ЧС	12
13	Казань ТЦ "Бехетле"	13
14	Казань ТЦ "ПаркХаус" II	14
15	Калуга ТРЦ "XXI век"	15
16	Коломна ТЦ "Рио"	16
17	Красноярск ТЦ "Взлетка Плаза"	17
18	Красноярск ТЦ "Июнь"	18
19	Курск ТЦ "Пушкинский"	19
20	Москва "Распродажа"	20
21	Москва МТРЦ "Афи Молл"	21
22	Москва Магазин С21	22
23	Москва ТК "Буденовский" (пав.А2)	23
24	Москва ТК "Буденовский" (пав.К7)	24
25	Москва ТРК "Атриум"	25
26	Москва ТЦ "Ареал" (Беляево)	26
27	Москва ТЦ "МЕГА Белая Дача II"	27
28	Москва ТЦ "МЕГА Теплый Стан" II	28
29	Москва ТЦ "Новый век" (Новокосино)	29
30	Москва ТЦ "Перловский"	30
31	Москва ТЦ "Семеновский"	31
32	Москва ТЦ "Серебряный Дом"	32
33	Мытищи ТРК "XL-3"	33

	shop_name	shop_id
34	Н.Новгород ТРЦ "РИО"	34
35	Н.Новгород ТРЦ "Фантастика"	35
36	Новосибирск ТРЦ "Галерея Новосибирск"	36
37	Новосибирск ТЦ "Мега"	37
38	Омск ТЦ "Мега"	38
39	РостовНаДону ТРК "Мегацентр Горизонт"	39
40	РостовНаДону ТРК "Мегацентр Горизонт" Островной	40
41	РостовНаДону ТЦ "Мега"	41
42	СПб ТК "Невский Центр"	42
43	СПб ТК "Сенная"	43
44	Самара ТЦ "Мелодия"	44
45	Самара ТЦ "ПаркХаус"	45
46	Сергиев Посад ТЦ "7Я"	46
47	Сургут ТРЦ "Сити Молл"	47
48	Томск ТРЦ "Изумрудный Город"	48
49	Тюмень ТРЦ "Кристалл"	49
50	Тюмень ТЦ "Гудвин"	50
51	Тюмень ТЦ "Зеленый Берег"	51
52	Уфа ТК "Центральный"	52
53	Уфа ТЦ "Семья" 2	53
54	Химки ТЦ "Мега"	54
55	Цифровой склад 1С-Онлайн	55
56	Чехов ТРЦ "Карнавал"	56
57	Якутск Орджоникидзе, 56	57
58	Якутск ТЦ "Центральный"	58
59	Ярославль ТЦ "Альтаир"	59

In [67]:
```
# getting rid of "!" before shop_names
shops['shop_name'] = shops['shop_name'].map(lambda x: x.split('!')[1] if x.startswith('!') else x)
shops['shop_name'] = shops["shop_name"].map(lambda x: 'СергиевПосад ТЦ "7Я"' if x == 'Сергиев Посад ТЦ "7Я"' else x)
```

In [68]:
```
#Extracting city names with Citry codes
shops['city'] = shops['shop_name'].map(lambda x: x.split(" ")[0])
# lets assign code to these city names too
shops['city_code'] = shops['city'].factorize()[0]
```

In [69]:
```
shops.head(2)
```

Out[69]:

	shop_name	shop_id	city	city_code
0	Якутск Орджоникидзе, 56 фран	0	Якутск	0
1	Якутск ТЦ "Центральный" фран	1	Якутск	0

In [70]:
```
for shop_id in shops['shop_id'].unique():
    shops.loc[shop_id, 'num_products'] = train[train['shop_id'] == shop_id]['item_id'].nunique()
    shops.loc[shop_id, 'min_price'] = train[train['shop_id'] == shop_id]['item_price'].min()
    shops.loc[shop_id, 'max_price'] = train[train['shop_id'] == shop_id]['item_price'].max()
    shops.loc[shop_id, 'mean_price'] = train[train['shop_id'] == shop_id]['item_price'].mean()
```

In [71]:
```
shops.head(2)
```

Out[71]:

	shop_name	shop_id	city	city_code	num_products	min_price	max_price	mean_price
0	Якутск Орджоникидзе, 56 фран	0	Якутск	0	3600.0	13.0	15653.0	563.444151
1	Якутск ТЦ "Центральный" фран	1	Якутск	0	2523.0	13.0	15653.0	515.350652

Processing Item Category data

The Item category name is designed like below:

- Item category name = type of the category + sub types

for example: an item category name **Служебные - Билеты ** is tranlated as **Service - Tickets**

where the `type` of this category is **Service** and `subtype` is **Tickets** (what kind of service)..

we will now add these new features to our dataset

In [72]:
```
item_cat
```

Out[72]:

	item_category_name	item_category_id
0	PC - Гарнитуры/Наушники	0
1	Аксессуары - PS2	1
2	Аксессуары - PS3	2
3	Аксессуары - PS4	3
4	Аксессуары - PSP	4
...	...	...
79	Служебные	79
80	Служебные - Билеты	80
81	Чистые носители (шпиль)	81
82	Чистые носители (штучные)	82
83	Элементы питания	83

84 rows × 2 columns

In [73]:
```
cat_list = []
for name in item_cat['item_category_name']:
    cat_list.append(name.split('-'))
```

creating a column `split` after `item_category_name` at '-'

In [74]:
```
item_cat['split'] = (cat_list)
item_cat['cat_type'] = item_cat['split'].map(lambda x: x[0])
item_cat['cat_type_code'] = item_cat['cat_type'].factorize()[0]
item_cat['sub_cat_type'] = item_cat['split'].map(lambda x: x[1] if len(x)>1 else x[0])
item_cat['sub_cat_type_code'] = item_cat['sub_cat_type'].factorize()[0]
```

In [75]:
```
item_cat.head(2)
```

Out[75]:

	item_category_name	item_category_id	split	cat_type	cat_type_code	sub_cat_type	sub_cat_type_code
0	PC - Гарнитуры/Наушники	0	[PC , Гарнитуры/Наушники]	PC	0	Гарнитуры/Наушники	0
1	Аксессуары - PS2	1	[Аксессуары , PS2]	Аксессуары	1	PS2	1

In [76]:
```
item_cat.drop('split', axis = 1, inplace=True)
item_cat.head(2)
```

Out[76]:

	item_category_name	item_category_id	cat_type	cat_type_code	sub_cat_type	sub_cat_type_code
0	PC - Гарнитуры/Наушники	0	PC	0	Гарнитуры/Наушники	0
1	Аксессуары - PS2	1	Аксессуары	1	PS2	1

Model creation and Prediction

Since we want to predict the sale for Nov, 2015 i.e. date_block_num = 34 so we will have to add that to our test to be able to make sale prediciton.

First lets create test and train dataset copies

In [77]:
```
train = train[train["item_cnt_day"]>0]
train = train[["month", "date_block_num", "shop_id", "item_id", "item_price", "item_cnt_day"]].groupby(
    ["date_block_num", "shop_id", "item_id"]).agg(
    {"item_price": "mean","item_cnt_day": "sum", "month": "min"}).reset_index()
train.rename(columns={"item_cnt_day": "item_cnt_month"}, inplace=True)
train = pd.merge(train, items, on="item_id", how="inner")
train = pd.merge(train, shops, on="shop_id", how="inner")
train = pd.merge(train, item_cat, on="item_category_id", how="inner")
```

In [78]:
```
train.head(2)
```

Out[78]:

	date_block_num	shop_id	item_id	item_price	item_cnt_month	month	item_name	item_category_id	shop_name	city	city_code	num_products	min_price	max_price	mean_price	item_category_name	cat_type	cat_type_
0	0	0	32	221.0	6.0	1	1+1	40	Якутск Орджоникидзе, 56 фран	Якутск	0	3600.0	13.0	15653.0	563.444151	Кино - DVD	Кино	
1	1	0	32	221.0	10.0	2	1+1	40	Якутск Орджоникидзе, 56 фран	Якутск	0	3600.0	13.0	15653.0	563.444151	Кино - DVD	Кино	

In [79]:
```
train.drop(['item_name', 'shop_name', 'city', 'item_category_name', 'cat_type', 'sub_cat_type'], axis = 1, inplace=True)
```

In [80]:
```
train.head(1)
```

Out[80]:

	date_block_num	shop_id	item_id	item_price	item_cnt_month	month	item_category_id	city_code	num_products	min_price	max_price	mean_price	cat_type_code	sub_cat_type_code
0	0	0	32	221.0	6.0	1	40	0	3600.0	13.0	15653.0	563.444151	11	22

In [81]:
```
train.head()
```

Out[81]:

	date_block_num	shop_id	item_id	item_price	item_cnt_month	month	item_category_id	city_code	num_products	min_price	max_price	mean_price	cat_type_code	sub_cat_type_code
0	0	0	32	221.0	6.0	1	40	0	3600.0	13.0	15653.0	563.444151	11	22
1	1	0	32	221.0	10.0	2	40	0	3600.0	13.0	15653.0	563.444151	11	22
2	0	0	35	247.0	1.0	1	40	0	3600.0	13.0	15653.0	563.444151	11	22
3	1	0	35	247.0	14.0	1	40	0	3600.0	13.0	15653.0	563.444151	11	22
4	0	0	43	221.0	1.0	1	40	0	3600.0	13.0	15653.0	563.444151	11	22

In [82]:
```
test_items.head()
```

Out[82]:

	ID	shop_id	item_id
0	0	5	5037
1	1	5	5320
2	2	5	5233
3	3	5	5232
4	4	5	5268

In [83]:
```
test_items.shape
```

Out[83]: (214200, 3)

In [84]:
```
train.shape
```

Out[84]: (1608223, 14)

In [85]:
```
train = train[train['shop_id'].isin(test_items['shop_id'].unique())]
train = train[train['item_id'].isin(test_items['item_id'].unique())]
```

In [86]:
```
train.shape
```

Out[86]: (599911, 14)

In [87]:
```
train.head(2)
```

Out[87]:

	date_block_num	shop_id	item_id	item_price	item_cnt_month	month	item_category_id	city_code	num_products	min_price	max_price	mean_price	cat_type_code	sub_cat_type_code
2833	0	3	32	349.0	3.0	1	40	2	5261.0	0.1	42990.0	1033.090115	11	22
2834	2	3	32	349.0	2.0	2	40	2	5261.0	0.1	42990.0	1033.090115	11	22

Since we want to predict the sale for Nov, 2015 i.e. `date_block_num = 34` so we will have to add that to our `test_dataset` to be able to make sale prediciton.

First lets create `test` and `train` dataset copies

In [88]:
```
final_train_dataset = train.copy()
final_test_dataset = test_items.copy()
```

In [89]:
```
import itertools
def data_preprocess(sales_train, test=None):
    indexlist = []
    for i in sales_train.date_block_num.unique():
        x = itertools.product(
            [i],
            sales_train.loc[sales_train.date_block_num == i].shop_id.unique(),
            sales_train.loc[sales_train.date_block_num == i].item_id.unique(),
        )
        indexlist.append(np.array(list(x)))
    df = pd.DataFrame(
        data=np.concatenate(indexlist, axis=0),
        columns=["date_block_num", "shop_id", "item_id"],
    )

    # Adding new revenue column
    sales_train["item_revenue_day"] = sales_train["item_price"] * sales_train["item_cnt_month"]
    # Aggregate item_id / shop_id item_cnts and revenue at the month level
    sales_train_grouped = sales_train.groupby(["date_block_num", "shop_id", "item_id"]).agg(
        item_cnt_month=pd.NamedAgg(column="item_cnt_month", aggfunc="sum"),
        item_revenue_month=pd.NamedAgg(column="item_revenue_day", aggfunc="sum"),
    )
    #print(sales_train_grouped)
    # Merge the grouped data with the index
    df = df.merge(
        sales_train_grouped, how="left", on=["date_block_num", "shop_id", "item_id"],
    )

    if test is not None:
        test["date_block_num"] = 34
        test["date_block_num"] = test["date_block_num"].astype(np.int8)
        test["shop_id"] = test.shop_id.astype(np.int8)
        test["item_id"] = test.item_id.astype(np.int16)
        test = test.drop(columns="ID")

        df = pd.concat([df, test[["date_block_num", "shop_id", "item_id"]]])

    # Fill empty item_cnt entries with 0
    df.item_cnt_month = df.item_cnt_month.fillna(0)
    df.item_revenue_month = df.item_revenue_month.fillna(0)

    return df

dataset_final = data_preprocess(final_train_dataset, final_test_dataset)
```

In [90]:
```
dataset_final = pd.merge(dataset_final, items, on="item_id", how="inner")
dataset_final = pd.merge(dataset_final, shops, on="shop_id", how="inner")
dataset_final = pd.merge(dataset_final, item_cat, on="item_category_id", how="inner")
dataset_final.head(3)
```

Out[90]:

	date_block_num	shop_id	item_id	item_cnt_month	item_revenue_month	item_name	item_category_id	shop_name	city	city_code	num_products	min_price	max_price	mean_price	item_category_name	cat_type	cat_ty
0	0	3	32	3.0	1047.0	1+1	40	Балашиха ТРК "Октябрь-Киномир"	Балашиха	2	5261.0	0.1	42990.0	1033.090115	Кино - DVD	Кино	
1	2	3	32	2.0	698.0	1+1	40	Балашиха ТРК "Октябрь-Киномир"	Балашиха	2	5261.0	0.1	42990.0	1033.090115	Кино - DVD	Кино	
2	4	3	32	1.0	349.0	1+1	40	Балашиха ТРК "Октябрь-Киномир"	Балашиха	2	5261.0	0.1	42990.0	1033.090115	Кино - DVD	Кино	

In [91]:
```
dataset_final.drop(['item_name', 'shop_name', 'city', 'item_category_name', 'cat_type', 'sub_cat_type'], axis = 1, inplace=True)
dataset_final.head(2)
```

Out[91]:

	date_block_num	shop_id	item_id	item_cnt_month	item_revenue_month	item_category_id	city_code	num_products	min_price	max_price	mean_price	cat_type_code	sub_cat_type_code
0	0	3	32	3.0	1047.0	40	2	5261.0	0.1	42990.0	1033.090115	11	22
1	2	3	32	2.0	698.0	40	2	5261.0	0.1	42990.0	1033.090115	11	22

In [92]:
```
dataset_final.shape
```

Out[92]: (2945738, 13)

Adding the lag feature for column names `item_cnt_month` and `item_revenue_month`

In [93]:
```
def lag_feature(matrix, lag_feature, lags):
    for lag in lags:
        newname = lag_feature + f"_lag_{lag}"
        print(f"Adding feature {newname}")
        targetseries = matrix.loc[:, ["date_block_num", "item_id", "shop_id"] + [lag_feature]]
        targetseries["date_block_num"] += lag
        targetseries = targetseries.rename(columns={lag_feature: newname})
        matrix = matrix.merge(
            targetseries, on=["date_block_num", "item_id", "shop_id"], how="left"
        )
#     print(matrix)
    return matrix

dataset_final = lag_feature(dataset_final, 'item_cnt_month', lags=[1,2,3])
dataset_final = lag_feature(dataset_final, 'item_revenue_month', lags=[1])
print("Lag features created..")
print(dataset_final.columns)
```

```
Adding feature item_cnt_month_lag_1
Adding feature item_cnt_month_lag_2
Adding feature item_cnt_month_lag_3
Adding feature item_revenue_month_lag_1
Lag features created..
Index(['date_block_num', 'shop_id', 'item_id', 'item_cnt_month',
       'item_revenue_month', 'item_category_id', 'city_code', 'num_products',
       'min_price', 'max_price', 'mean_price', 'cat_type_code',
       'sub_cat_type_code', 'item_cnt_month_lag_1', 'item_cnt_month_lag_2',
       'item_cnt_month_lag_3', 'item_revenue_month_lag_1'],
      dtype='object')
```

In [94]:
```
dataset_final.fillna(0, inplace= True)
dataset_final.head(2)
```

Out[94]:

	date_block_num	shop_id	item_id	item_cnt_month	item_revenue_month	item_category_id	city_code	num_products	min_price	max_price	mean_price	cat_type_code	sub_cat_type_code	item_cnt_month_lag_1	item_cnt_month
0	0	3	32	3.0	1047.0	40	2	5261.0	0.1	42990.0	1033.090115	11	22	0.0	
1	2	3	32	2.0	698.0	40	2	5261.0	0.1	42990.0	1033.090115	11	22	0.0	

leaving behind the sale record of year 2013

In [95]:
```
matrix = dataset_final[dataset_final.date_block_num>=12]
matrix.reset_index(drop=True, inplace=True)
```

In [96]:
```
1 matrix.head(2)
```

Out[96]:

	date_block_num	shop_id	item_id	item_cnt_month	item_revenue_month	item_category_id	city_code	num_products	min_price	max_price	mean_price	cat_type_code	sub_cat_type_code	item_cnt_month_lag_1	item_cnt_month
0	14	3	32	2.0	298.0	40	2	5261.0	0.1	42990.0	1033.090115	11	22	0.0	
1	15	3	32	1.0	149.0	40	2	5261.0	0.1	42990.0	1033.090115	11	22	2.0	

In [97]:
```
1 matrix.columns
```

Out[97]:
```
Index(['date_block_num', 'shop_id', 'item_id', 'item_cnt_month',
       'item_revenue_month', 'item_category_id', 'city_code', 'num_products',
       'min_price', 'max_price', 'mean_price', 'cat_type_code',
       'sub_cat_type_code', 'item_cnt_month_lag_1', 'item_cnt_month_lag_2',
       'item_cnt_month_lag_3', 'item_revenue_month_lag_1'],
      dtype='object')
```

Documentation

- final_train_df = train[['date_block_num','item_id','shop_id','item_cnt_month']]
- final_train_df = train.copy()
- final_train_df = pd.concat([final_train_df, test_copy[["date_block_num", "shop_id", "item_id"]]])
- final_train_df = final_train_df.pivot_table(index = ['shop_id','item_id'],values = ['item_cnt_month'],columns = ['date_block_num'],fill_value = 0,aggfunc='sum')
- final_train_df.reset_index(inplace = True)
- final_train_df = final_train_df.pivot_table(index=['item_id','shop_id'], columns = 'date_block_num', values = 'item_cnt_month', fill_value = 0).reset_index()
- final_train_df = pd.merge(test_items,final_train_df,on = ['item_id','shop_id'],how = 'left')
- final_train_df.fillna(0,inplace = True)
- final_train_df

Model

In [98]:

```
def fit_booster(
    X_train,
    y_train,
    X_test=None,
    y_test=None,
    params=None,
    test_run=False,
    categoricals=[],
    dropcols=[],
    early_stopping=True,
):
    if params is None:
        params = {"learning_rate": 0.1, "subsample_for_bin": 300000, "n_estimators": 50}

    early_stopping_rounds = None
    if early_stopping == True:
        early_stopping_rounds = 50

    if test_run:
        eval_set = [(X_train, y_train)]
    else:
        eval_set = [(X_train, y_train), (X_test, y_test)]

    booster = lgbm.LGBMRegressor(**params)

    categoricals = [c for c in categoricals if c in X_train.columns]

    booster.fit(
        X_train,
        y_train,
        eval_set=eval_set,
        eval_metric=["rmse"],
        verbose=100,
        categorical_feature=categoricals,
        early_stopping_rounds=early_stopping_rounds,
    )

    return booster



keep_from_month = 2  # The first couple of months are dropped because of distortions to their features (e.g. wrong item age)
test_month = 33
# dropping this will reduce overfitting
dropcols = [
    "shop_id",
    "item_id"
]
valid = matrix.drop(columns=dropcols).loc[matrix.date_block_num == test_month, :]
train__ = matrix.drop(columns=dropcols).loc[matrix.date_block_num < test_month, :]
train__ = train__[train__.date_block_num >= keep_from_month]
X_train = train__.drop(columns="item_cnt_month")
y_train = train__.item_cnt_month
X_valid = valid.drop(columns="item_cnt_month")
y_valid = valid.item_cnt_month

```

```
params = {
    "num_leaves": 966,
    "cat_smooth": 45.01680827234465,
    "min_child_samples": 27,
    "min_child_weight": 0.021144950289224463,
    "max_bin": 214,
    "learning_rate": 0.01,
    "subsample_for_bin": 300000,
    "min_data_in_bin": 7,
    "colsample_bytree": 0.8,
    "subsample": 0.6,
    "subsample_freq": 5,
    "n_estimators": 8000,
}

categoricals = [
    "item_category_id",
    "month",
]  # These features will be set as categorical features by LightGBM and handled differently

lgbooster = fit_booster(
    X_train,
    y_train,
    X_valid,
    y_valid,
    params=params,
    test_run=False,
    categoricals=categoricals,
)
```

```
[100]   training's rmse: 2.55394        training's l2: 6.52262  valid_1's rmse: 1.59872 valid_1's l2: 2.55591
[200]   training's rmse: 1.90449        training's l2: 3.62709  valid_1's rmse: 1.22228 valid_1's l2: 1.49397
[300]   training's rmse: 1.65013        training's l2: 2.72292  valid_1's rmse: 1.08416 valid_1's l2: 1.1754
[400]   training's rmse: 1.52857        training's l2: 2.33653  valid_1's rmse: 1.02781 valid_1's l2: 1.0564
[500]   training's rmse: 1.45359        training's l2: 2.11291  valid_1's rmse: 1.0132  valid_1's l2: 1.02657
[600]   training's rmse: 1.39651        training's l2: 1.95025  valid_1's rmse: 0.992475        valid_1's l2: 0.985007
[700]   training's rmse: 1.34316        training's l2: 1.80408  valid_1's rmse: 0.976694        valid_1's l2: 0.95393
[800]   training's rmse: 1.30699        training's l2: 1.70823  valid_1's rmse: 0.965576        valid_1's l2: 0.932337
[900]   training's rmse: 1.27072        training's l2: 1.61472  valid_1's rmse: 0.960486        valid_1's l2: 0.922533
[1000]  training's rmse: 1.23323        training's l2: 1.52086  valid_1's rmse: 0.946623        valid_1's l2: 0.896095
[1100]  training's rmse: 1.20694        training's l2: 1.4567   valid_1's rmse: 0.939684        valid_1's l2: 0.883005
[1200]  training's rmse: 1.18482        training's l2: 1.4038   valid_1's rmse: 0.934607        valid_1's l2: 0.87349
[1300]  training's rmse: 1.15897        training's l2: 1.34322  valid_1's rmse: 0.923659        valid_1's l2: 0.853145
[1400]  training's rmse: 1.13379        training's l2: 1.28548  valid_1's rmse: 0.918048        valid_1's l2: 0.842813
[1500]  training's rmse: 1.11201        training's l2: 1.23658  valid_1's rmse: 0.912015        valid_1's l2: 0.831771
[1600]  training's rmse: 1.08915        training's l2: 1.18624  valid_1's rmse: 0.903441        valid_1's l2: 0.816206
[1700]  training's rmse: 1.06966        training's l2: 1.14417  valid_1's rmse: 0.899769        valid_1's l2: 0.809585
```

Prediction

In [99]:
```
matrix['item_cnt_month'] = matrix['item_cnt_month'].clip(0,20)
keep_from_month = 2
test_month = 34
test__ = matrix.loc[matrix.date_block_num==test_month, :]
X_test = test__.drop(columns="item_cnt_month")
y_test = test__.item_cnt_month

X_test["item_cnt_month"] = lgbooster.predict(X_test.drop(columns=dropcols)).clip(0, 20)
```

Submission

In [100]:
```
testing = test_items.merge(
    X_test[["shop_id", "item_id", "item_cnt_month"]],
    on=["shop_id", "item_id"],
    how="inner",
    copy=True,
)
# Verify that the indices of the submission match the original
assert test_items.equals(testing[["ID", "shop_id", "item_id"]])
testing[["ID", "item_cnt_month"]].to_csv("./submission.csv", index=False)
```

FINISH