상점의 매출 미래 예측 (한달 기준)

- 대회 URL : https://www.kaggle.com/c/competitive-data-science-predict-future-sales (https://www.kaggle.com/c/competitive-data-science-predict-future-sales)
- 대회 설명 : Coursera '데이터 과학 대회에서 우승하는 방법' 과정의 최종 프로젝트
  - 러시아 최대 소프트웨어 회사 중 하나인 1C Company에서 제공된 데이터 셋
- 대회 예측 : 다음 달의 모든 제품 및 매장에 대한 총 매출을 예측해야 하는 과제

REF : https://www.kaggle.com/ashishpatel26/predict-sales-price-using-xgboost (https://www.kaggle.com/ashishpatel26/predict-sales-price-using-xgboost)

라이브러리 불러오기

In [1]:

```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_style("ticks", {"xtick.major.size": 8, "ytick.major.size": 8})
plt.style.use('ggplot')

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/competitive-data-science-predict-future-sales/items.csv
/kaggle/input/competitive-data-science-predict-future-sales/sample_submission.csv
/kaggle/input/competitive-data-science-predict-future-sales/item_categories.csv
/kaggle/input/competitive-data-science-predict-future-sales/sales_train.csv
/kaggle/input/competitive-data-science-predict-future-sales/shops.csv
/kaggle/input/competitive-data-science-predict-future-sales/test.csv
```

데이터 불러오기

파일명	내용	행열
sales_train.csv	학습 데이터. 2013년 1월부터 2015년 10월까지의 일일 기록 데이터	2935849행, 6열
test.csv	테스트 데이터. 상점과 제품의 2015년 11월 매출을 예측	214200행, 3열
items.csv	항목/제품에 대한 추가 정보	22170행, 3열
item_categories.csv	항목 카테고리에 대한 추가 정보	84행, 2열
shops.csv	상점에 대한 추가 정보	60행, 2열
sample_submission.csv	올바른 형식의 샘플 파일 제출	

In [2]:

```
train = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales/sales_train.csv')
test  = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales/test.csv')
items = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales/items.csv')
item_category = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales/item_catego
shops = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales/shops.csv')
sub = pd.read_csv("/kaggle/input/competitive-data-science-predict-future-sales/sample_submission.csv
```

구분	컬럼명	설명	값
train	date	dd / mm / yyyy 형식의 날짜	날짜 데이터
train	date_block_num	편의를 위해 사용되는 연속 월 번호입니다.	2013/01(1)~2015/10(33)
train	shop_id	상점 고유 ID	0~59
train	item_id	항목 ID	0~22169
train	item_price	상품의 현재 가격	-1~307980
train	item_cnt_day	판매 된 제품 수입니다. 이 측정 값의 월별 금액을 예측하고 있습니다.	-22~2169
test	ID	테스트 예측을 위한 ID	0~214199
test	shop_id	상점 고유 ID	2~59
test	item_id	항목 ID	30~22167
sub	ID	테스트 예측을 위한 ID	0~214199
sub	item_cnt_month	예측해야 하는 값	default:0.5
items	item_name	항목 이름	범주의 개수(22170) '! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.) D', '!ABBYY FineReader 12 Professional Edition Full [PC, Цифровая версия]'
items	item_id	항목 ID	0~22169
items	item_category_id	항목 카테고리의 고유 식별자	0~83
items_categories	item_category_name	항목 카테고리 이름	범주의 개수(84) 'PC - Гарнитуры/Наушники' 'Аксессуары - PS2' 'Аксессуары - PS3' 'Аксессуары - PS4'
items_categories	item_category_id	항목 카테고리의 고유 식별자	0~83
shops	shop_name	상점 이름	범주의 개수(60)
shops	shop_id	상점 고유 ID	0~59

In [3]:

```
# 행열
print(train.shape, train.columns)
print(test.shape, test.columns)
print(items.shape, items.columns)
print(item_category.shape, item_category.columns)
print(shops.shape, shops.columns)
print(shops.shape, shops.columns)
```

```
(2935849, 6) Index(['date', 'date_block_num', 'shop_id', 'item_id', 'item_price',
       'item_cnt_day'],
      dtype='object')
(214200, 3) Index(['ID', 'shop_id', 'item_id'], dtype='object')
(22170, 3) Index(['item_name', 'item_id', 'item_category_id'], dtype='object')
(84, 2) Index(['item_category_name', 'item_category_id'], dtype='object')
(60, 2) Index(['shop_name', 'shop_id'], dtype='object')
(60, 2) Index(['shop_name', 'shop_id'], dtype='object')
```

데이터 탐색

In [4]:

```
def eda(data):
    print("----------Top-5----------")
    print(data.head(5))
    print("-----------데이터 셋 구조-----------")
    print(data.info())
    print("---------- 결측치 확인 -----------")
    print(data.isnull().sum())
    print("----------Null 값 확인-----------")
    print(data.isna().sum())
    print("----------데이터 행열 ----------")
    print(data.shape)
```

그래프를 통한 인사이트 얻기

In [5]:

```
def graph_insight(data):
    print(set(data.dtypes.tolist()))
    df_num = data.select_dtypes(include = ['float64', 'int64'])
    df_num.hist(figsize=(16, 16), bins=50, xlabelsize=8, ylabelsize=8);
```

데이터 중복 제거

In [6]:

```
def drop_duplicate(data, subset):
    print('(처리전) 데이터 행열:', data.shape)
    before = data.shape[0]
    data.drop_duplicates(subset,keep='first', inplace=True)
    data.reset_index(drop=True, inplace=True)
    print('(처리후) 데이터 행열:', data.shape)
    after = data.shape[0]
    print('Total Duplicate:', before-after)
```

In [7]:

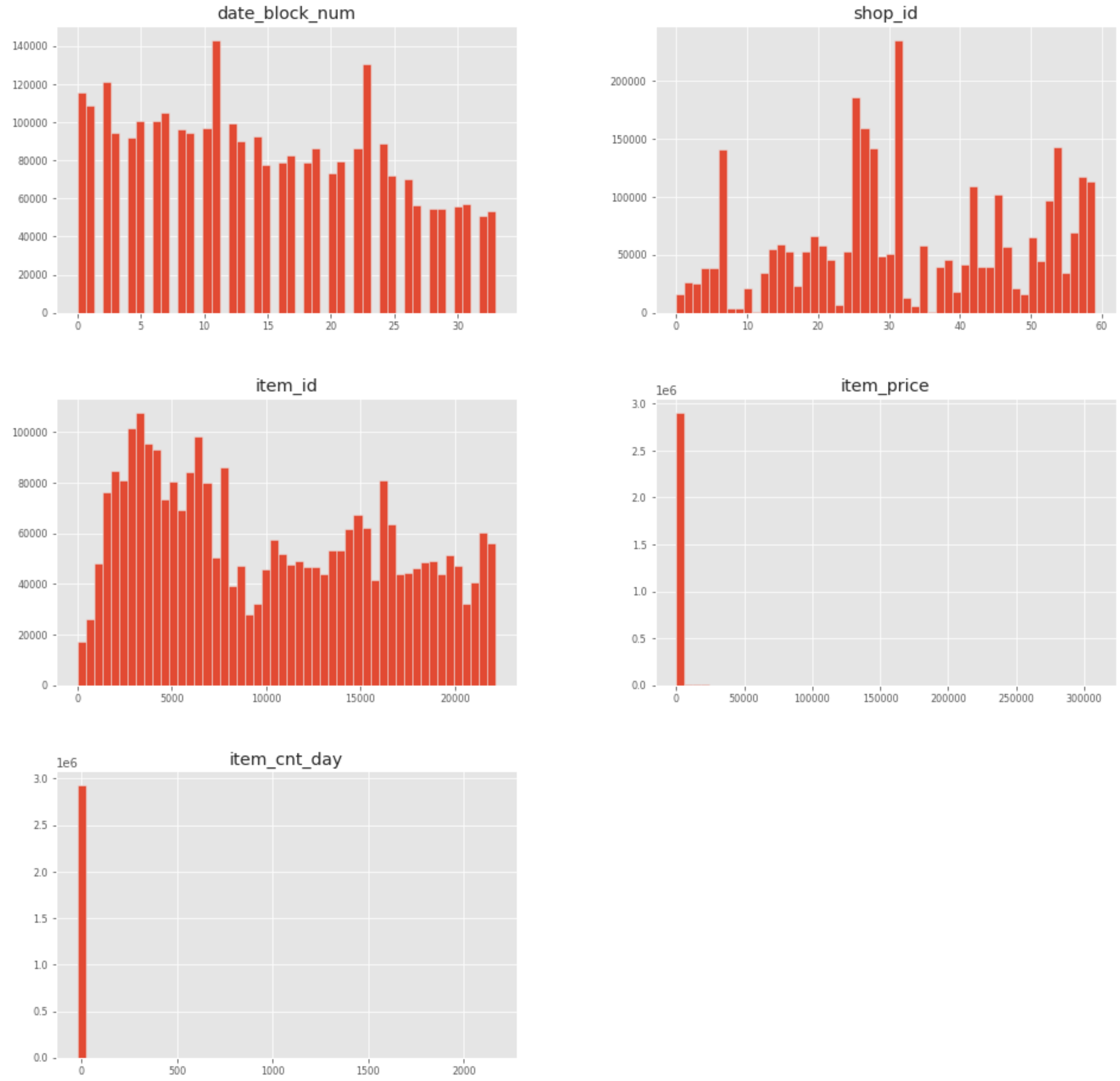
```
eda(train)
```

```
----------Top-5----------
         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
-----------데이터 셋 구조-----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2935849 entries, 0 to 2935848
Data columns (total 6 columns):
 #   Column          Dtype  
---  ------          -----  
 0   date            object 
 1   date_block_num  int64  
 2   shop_id         int64  
 3   item_id         int64  
 4   item_price      float64
 5   item_cnt_day    float64
dtypes: float64(2), int64(3), object(1)
memory usage: 134.4+ MB
None
---------- 결측치 확인 -----------
date              0
date_block_num    0
shop_id           0
item_id           0
item_price        0
item_cnt_day      0
dtype: int64
----------Null 값 확인-----------
date              0
date_block_num    0
shop_id           0
item_id           0
item_price        0
item_cnt_day      0
dtype: int64
----------데이터 행열 ----------
(2935849, 6)
```

In [8]:

```
graph_insight(train)
```

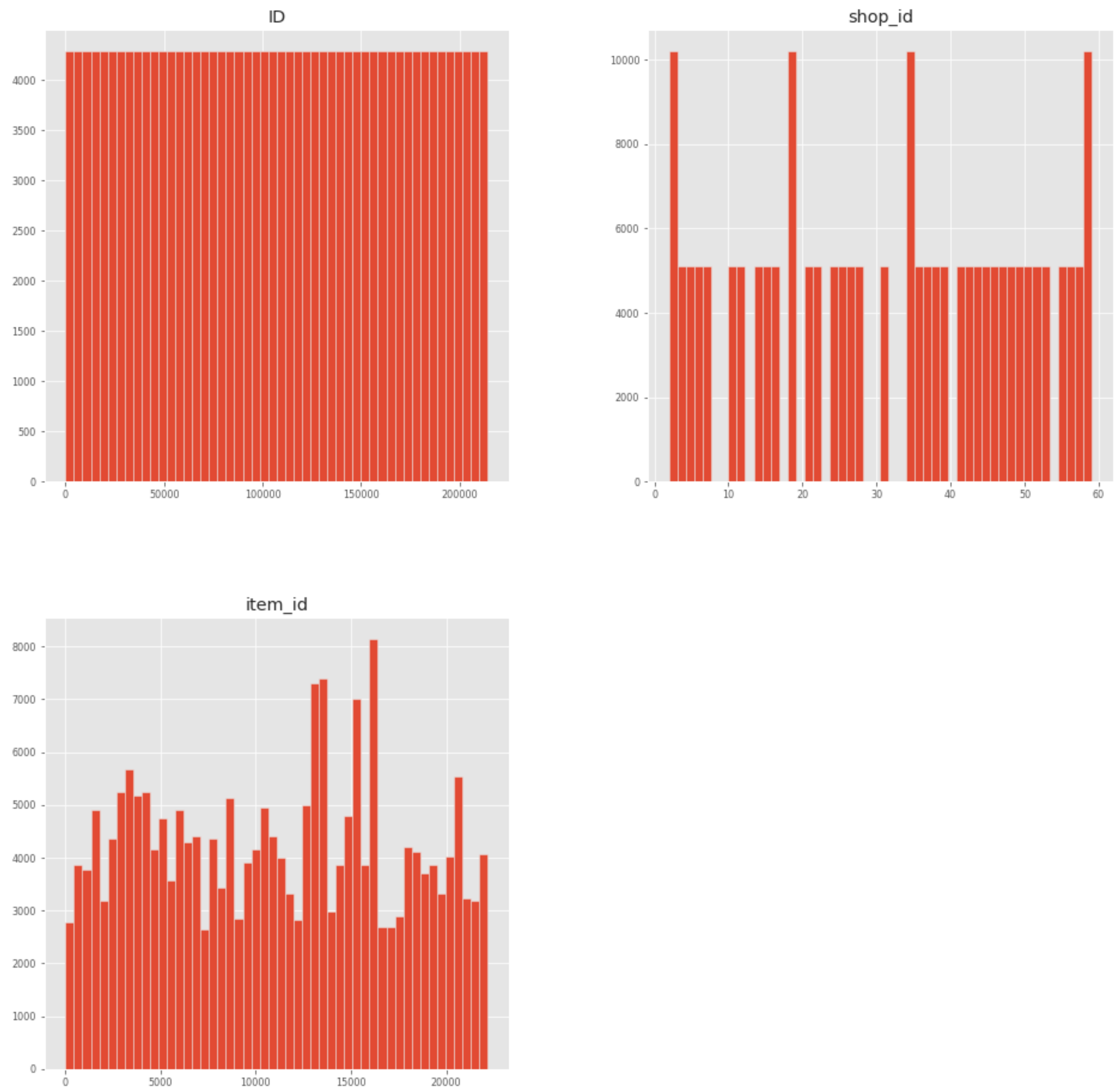
```
{dtype('int64'), dtype('O'), dtype('float64')}
```



In [9]:

```
graph_insight(test)
```

```
{dtype('int64')}
```



In [10]:

```
### 중복 데이터 제거
subset = ['date', 'date_block_num', 'shop_id', 'item_id','item_cnt_day']
drop_duplicate(train, subset = subset)
```

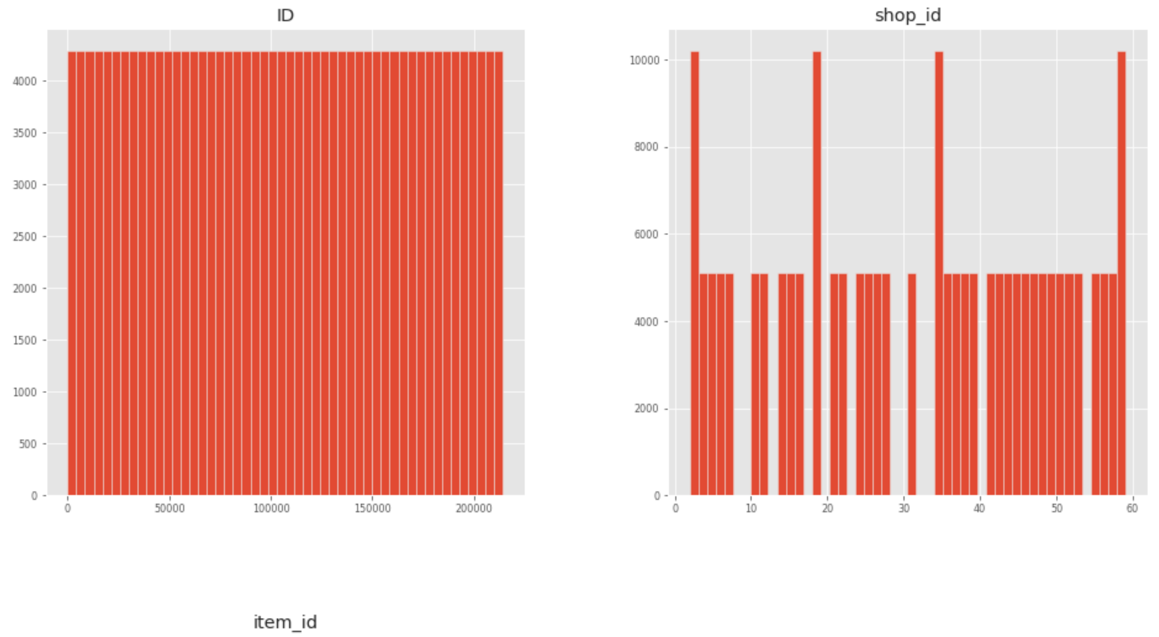
```
(처리전) 데이터 행열: (2935849, 6)
(처리후) 데이터 행열: (2935825, 6)
Total Duplicate: 24
```

테스트 데이터

In [11]:

```
# test insight
eda(test)
graph_insight(test)
```

```
----------Top-5----------
   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
-----------데이터 셋 구조-----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214200 entries, 0 to 214199
Data columns (total 3 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   ID       214200 non-null  int64
 1   shop_id  214200 non-null  int64
 2   item_id  214200 non-null  int64
dtypes: int64(3)
memory usage: 4.9 MB
None
---------- 결측치 확인 -----------
ID         0
shop_id    0
item_id    0
dtype: int64
----------Null 값 확인-----------
ID         0
shop_id    0
item_id    0
dtype: int64
----------데이터 행열 ----------
(214200, 3)
{dtype('int64')}
```

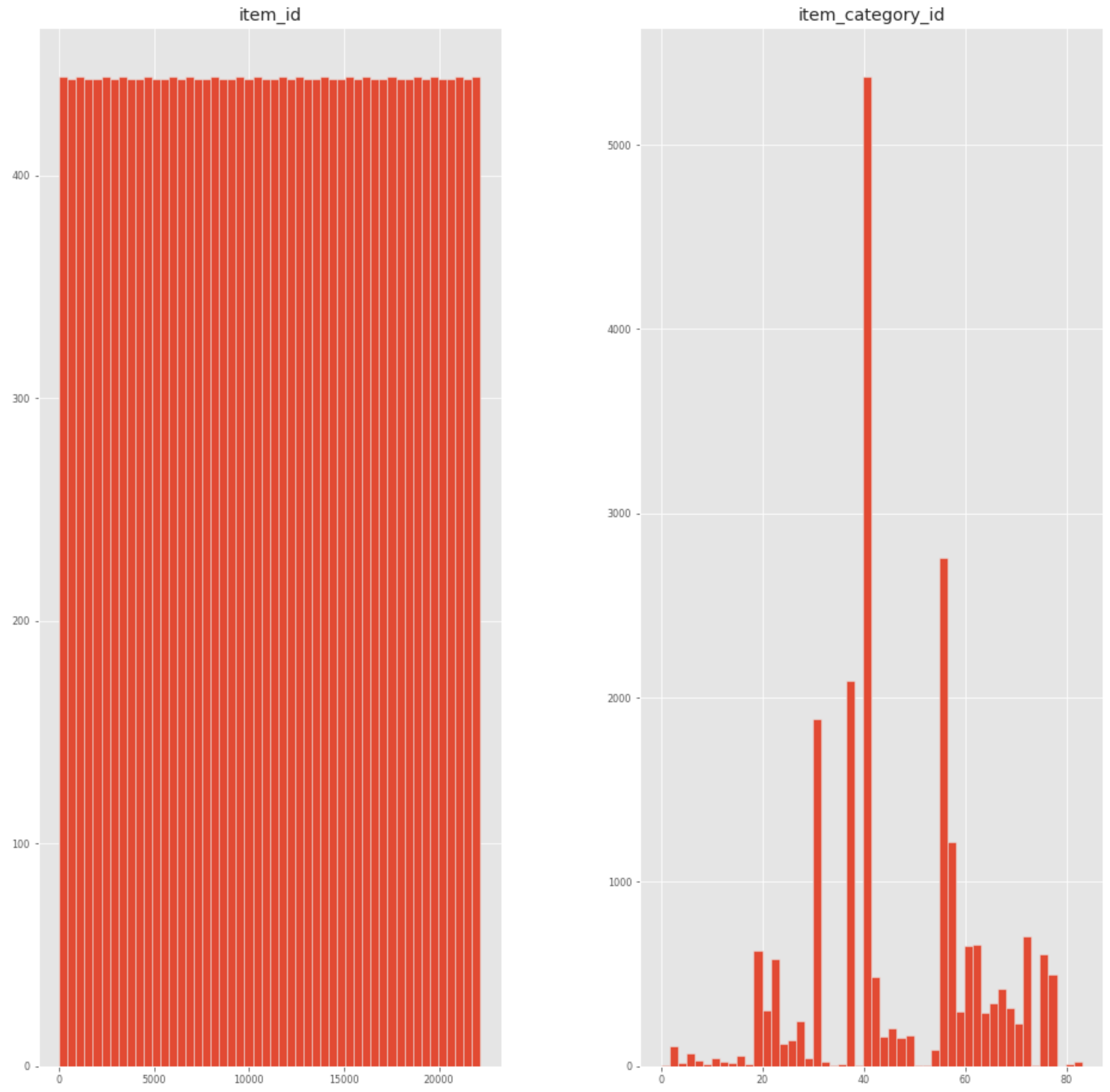

ID
0
500
1000
1500
2000
2500
3000
3500
4000
0
50000
100000
150000
200000
shop_id
0
2000
4000
6000
8000
10000
0
10
20
30
40
50
60
item_id

item 데이터

In [12]:

```
eda(items)
graph_insight(items)
```

```
----------Top-5----------
                                    item_name  item_id  ₩
0          ! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.)         D        0
1  !ABBYY FineReader 12 Professional Edition Full...        1
2      ***В ЛУЧАХ СЛАВЫ   (UNV)                    D        2
3    ***ГОЛУБАЯ ВОЛНА  (Univ)                      D        3
4        ***КОРОБКА (СТЕКЛО)                       D        4

   item_category_id
0                40
1                76
2                40
3                40
4                40
-----------데이터 셋 구조-----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22170 entries, 0 to 22169
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   item_name         22170 non-null  object
 1   item_id           22170 non-null  int64 
 2   item_category_id  22170 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 519.7+ KB
None
---------- 결측치 확인 -----------
item_name           0
item_id             0
item_category_id    0
dtype: int64
----------Null 값 확인-----------
item_name           0
item_id             0
item_category_id    0
dtype: int64
----------데이터 행열 ----------
(22170, 3)
{dtype('int64'), dtype('O')}
```



item category

In [13]:

```
eda(item_category)
```

```
-----------Top-5----------
        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
------------데이터 셋 구조-----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 2 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   item_category_name  84 non-null     object
 1   item_category_id    84 non-null     int64 
dtypes: int64(1), object(1)
memory usage: 1.4+ KB
None
---------- 결측치 확인 -----------
item_category_name    0
item_category_id      0
dtype: int64
----------Null 값 확인-----------
item_category_name    0
item_category_id      0
dtype: int64
----------데이터 행열 ----------
(84, 2)
```

shops 데이터

In [14]:

```
eda(shops)
```

```
----------Top-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
-----------데이터 셋 구조-----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   shop_name  60 non-null     object
 1   shop_id    60 non-null     int64 
dtypes: int64(1), object(1)
memory usage: 1.1+ KB
None
---------- 결측치 확인 -----------
shop_name    0
shop_id      0
dtype: int64
----------Null 값 확인-----------
shop_name    0
shop_id      0
dtype: int64
----------데이터 행열 ----------
(60, 2)
```

In [15]:

```
### 함수 - 데이터의 최대, 최소 및 통계량
def unresanable_data(data):
    print("Min Value:",data.min())
    print("Max Value:",data.max())
    print("Average Value:",data.mean())
    print("Center Point of Data:",data.median())
```

In [16]:

```
train.describe()
```

Out[16]:

	date_block_num	shop_id	item_id	item_price	item_cnt_day
count	2.935825e+06	2.935825e+06	2.935825e+06	2.935825e+06	2.935825e+06
mean	1.456992e+01	3.300171e+01	1.019721e+04	8.908558e+02	1.242643e+00
std	9.422984e+00	1.622699e+01	6.324298e+03	1.729806e+03	2.618845e+00
min	0.000000e+00	0.000000e+00	0.000000e+00	-1.000000e+00	-2.200000e+01
25%	7.000000e+00	2.200000e+01	4.476000e+03	2.490000e+02	1.000000e+00
50%	1.400000e+01	3.100000e+01	9.343000e+03	3.990000e+02	1.000000e+00
75%	2.300000e+01	4.700000e+01	1.568400e+04	9.990000e+02	1.000000e+00
max	3.300000e+01	5.900000e+01	2.216900e+04	3.079800e+05	2.169000e+03

아웃라이어(outliers) - 이상치

- item_price에 0이하와 300000이상은 제외시키자.

In [17]:

```
# -1 and 307980 looks like outliers
print('before train shape:', train.shape)
train = train[(train.item_price > 0) & (train.item_price < 300000)]
print('after train shape:', train.shape)
```

```
before train shape: (2935825, 6)
after train shape: (2935823, 6)
```

매달의 매출액 합 구하기

In [18]:

```
train.groupby('date_block_num').sum().head()
```

Out[18]:

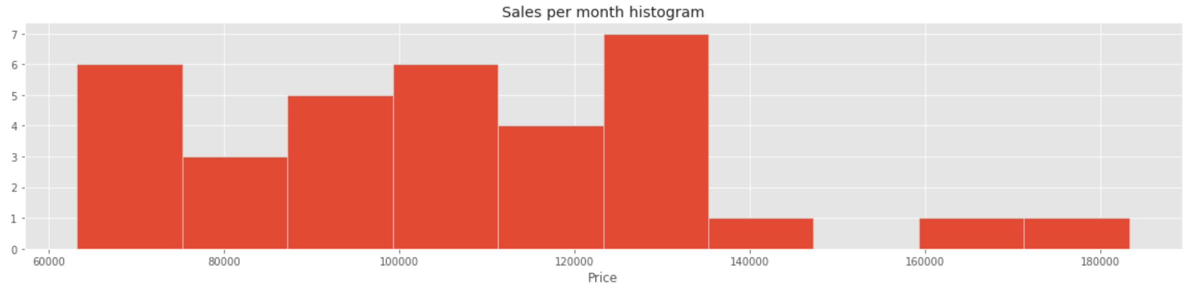
	shop_id	item_id	item_price	item_cnt_day
date_block_num				
0	3416958	1183925474	8.221009e+07	131476.0
1	3111541	1076016145	7.557875e+07	128088.0
2	4016391	1220887356	8.429603e+07	147140.0
3	3164924	971331915	6.151247e+07	107189.0
4	3093967	950370015	5.727413e+07	106969.0

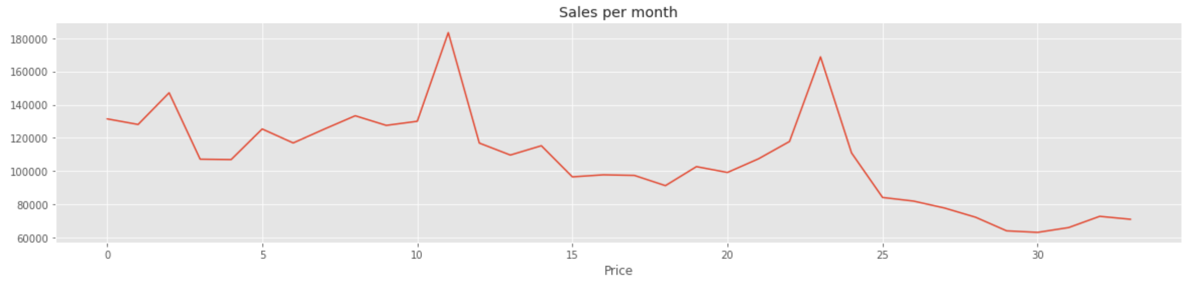
In [19]:

```
# 월별 매출
train.groupby('date_block_num').sum()['item_cnt_day'].hist(figsize = (20,4))
plt.title('Sales per month histogram')
plt.xlabel('Price')

plt.figure(figsize = (20,4))
sns.lineplot(data=train.groupby('date_block_num').sum()['item_cnt_day'])
plt.title('Sales per month')
plt.xlabel('Price')
```

Out[19]:

```
Text(0.5, 0, 'Price')
```





데이터 분포 확인

In [20]:

```
train.item_price.value_counts().sort_index(ascending=False)
```

Out[20]:

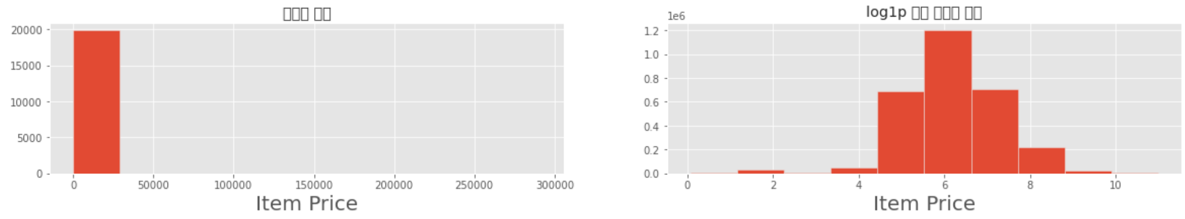
```
59200.0000       1
50999.0000       1
49782.0000       1
42990.0000       4
42000.0000       1
              ...
0.2000           1
0.1000        2932
0.0900           1
0.0875           1
0.0700           2
Name: item_price, Length: 19991, dtype: int64
```

In [21]:

```
unresanable_data(train['item_price'])  # 통계량

# 상품 가격으로 정렬
count_price = train.item_price.value_counts().sort_index(ascending=False)

plt.subplot(221)
count_price.hist(figsize=(20,6))
plt.xlabel('Item Price', fontsize=20);
plt.title('데이터 분포')

plt.subplot(222)
train.item_price.map(np.log1p).hist(figsize=(20,6))
plt.xlabel('Item Price', fontsize=20);
plt.title('log1p 변환 데이터 분포')
train.loc[:,'item_price'] = train.item_price.map(np.log1p)
```

```
Min Value: 0.07
Max Value: 59200.0
Average Value: 890.7514892291379
Center Point of Data: 399.0

/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:214: Runti
meWarning: Glyph 45936 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:214: Runti
meWarning: Glyph 51060 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:214: Runti
meWarning: Glyph 53552 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:214: Runti
meWarning: Glyph 48516 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:214: Runti
meWarning: Glyph 54252 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:214: Runti
meWarning: Glyph 48320 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:214: Runti
meWarning: Glyph 54872 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:183: Runti
meWarning: Glyph 45936 missing from current font.
  font.set_text(s, 0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:183: Runti
meWarning: Glyph 51060 missing from current font.
  font.set_text(s, 0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:183: Runti
meWarning: Glyph 53552 missing from current font.
  font.set_text(s, 0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:183: Runti
meWarning: Glyph 48516 missing from current font.
  font.set_text(s, 0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:183: Runti
meWarning: Glyph 54252 missing from current font.
  font.set_text(s, 0, flags=flags)
```

```
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:183: Runti
meWarning: Glyph 48320 missing from current font.
  font.set_text(s, 0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:183: Runti
meWarning: Glyph 54872 missing from current font.
  font.set_text(s, 0, flags=flags)
```



In [22]:

```
print( train.date_block_num.unique() )
print( train.shop_id.unique() )
print( train.item_id.unique() )
```

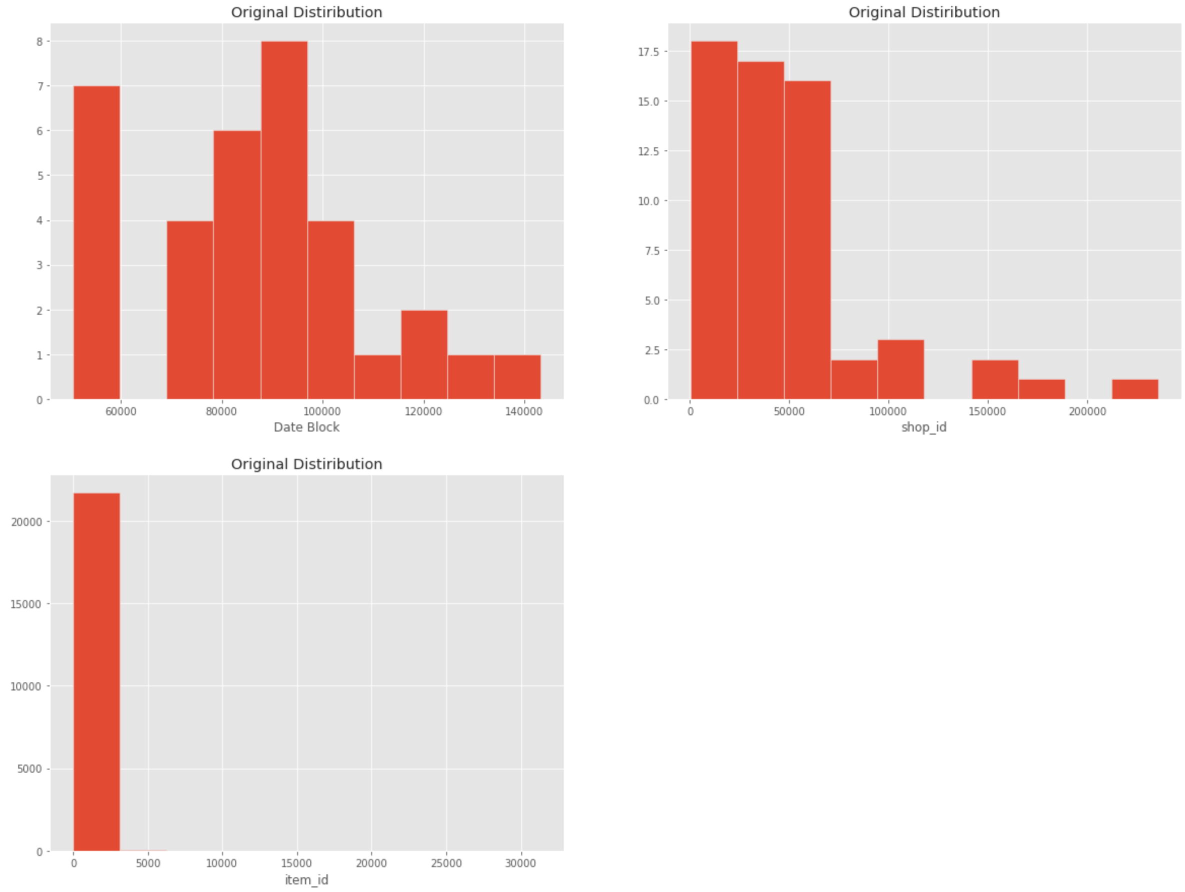
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33]
[59 25 24 23 19 22 18 21 28 27 29 26  4  6  2  3  7  0  1 16 15  8 10 14
 13 12 53 31 30 32 35 56 54 47 50 42 43 52 51 41 38 44 37 46 45  5 57 58
 55 17  9 49 39 40 48 34 33 20 11 36]
[22154  2552  2554 ...  7610  7635  7640]
```

In [23]:

```
# unresanable_data(train['date_block_num'])

### Data Block와 shop_id, item_id의 값에 대한 개수의 그래프
count_price = train.date_block_num.value_counts().sort_index(ascending=False)
plt.subplot(221)
count_price.hist(figsize=(20,15))
plt.xlabel('Date Block');
plt.title('Original Distiribution')

count_price = train.shop_id.value_counts().sort_index(ascending=False)
plt.subplot(222)
count_price.hist(figsize=(20,15))
plt.xlabel('shop_id');
plt.title('Original Distiribution')

count_price = train.item_id.value_counts().sort_index(ascending=False)
plt.subplot(223)
count_price.hist(figsize=(20,15))
plt.xlabel('item_id');
plt.title('Original Distiribution')
```

Out[23]:

```
Text(0.5, 1.0, 'Original Distiribution')
```



In [24]:

```
list(item_category.item_category_name)
```

Out[24]:

```
['PC - Гарнитуры/Наушники',
 'Аксессуары - PS2',
 'Аксессуары - PS3',
 'Аксессуары - PS4',
 'Аксессуары - PSP',
 'Аксессуары - PSVita',
 'Аксессуары - XBOX 360',
 'Аксессуары - XBOX ONE',
 'Билеты (Цифра)',
 'Доставка товара',
 'Игровые консоли - PS2',
 'Игровые консоли - PS3',
 'Игровые консоли - PS4',
 'Игровые консоли - PSP',
 'Игровые консоли - PSVita',
 'Игровые консоли - XBOX 360',
 'Игровые консоли - XBOX ONE',
 'Игровые консоли - Прочие',
 'Игры - PS2',
 'Игры - PS3',
 'Игры - PS4',
 'Игры - PSP',
 'Игры - PSVita',
 'Игры - XBOX 360',
 'Игры - XBOX ONE',
 'Игры - Аксессуары для игр',
 'Игры Android - Цифра',
 'Игры MAC - Цифра',
 'Игры PC - Дополнительные издания',
 'Игры PC - Коллекционные издания',
 'Игры PC - Стандартные издания',
 'Игры PC - Цифра',
 'Карты оплаты (Кино, Музыка, Игры)',
 'Карты оплаты - Live!',
 'Карты оплаты - Live! (Цифра)',
 'Карты оплаты - PSN',
 'Карты оплаты - Windows (Цифра)',
 'Кино - Blu-Ray',
 'Кино - Blu-Ray 3D',
 'Кино - Blu-Ray 4K',
 'Кино - DVD',
 'Кино - Коллекционное',
 'Книги - Артбуки, энциклопедии',
 'Книги - Аудиокниги',
 'Книги - Аудиокниги (Цифра)',
 'Книги - Аудиокниги 1С',
 'Книги - Бизнес литература',
 'Книги - Комиксы, манга',
 'Книги - Компьютерная литература',
 'Книги - Методические материалы 1С',
 'Книги - Открытки',
 'Книги - Познавательная литература',
 'Книги - Путеводители',
 'Книги - Художественная литература',
 'Книги - Цифра',
 'Музыка - CD локального производства',
 'Музыка - CD фирменного производства',
 'Музыка - MP3',
 'Музыка - Винил',
 'Музыка - Музыкальное видео',
 'Музыка - Подарочные издания',
 'Подарки - Атрибутика',
 'Подарки - Гаджеты, роботы, спорт',
 'Подарки - Мягкие игрушки',
 'Подарки - Настольные игры',
 'Подарки - Настольные игры (компактные)',
 'Подарки - Открытки, наклейки',
 'Подарки - Развитие',
 'Подарки - Сертификаты, услуги',
 'Подарки - Сувениры',
 'Подарки - Сувениры (в навеску)',
 'Подарки - Сумки, Альбомы, Коврики д/мыши',
 'Подарки - Фигурки',
 'Программы - 1С:Предприятие 8',
 'Программы - MAC (Цифра)',
 'Программы - Для дома и офиса',
 'Программы - Для дома и офиса (Цифра)',
 'Программы - Обучающие',
 'Программы - Обучающие (Цифра)',
 'Служебные',
 'Служебные - Билеты',
 'Чистые носители (шпиль)',
 'Чистые носители (штучные)',
 'Элементы питания']
```

아이템 카테고리 이름를 영어로 변경

In [25]:

```
l = list(item_category.item_category_name)
l_cat = l

for ind in range(1,8):
    l_cat[ind] = 'Access'

for ind in range(10,18):
    l_cat[ind] = 'Consoles'

for ind in range(18,25):
    l_cat[ind] = 'Consoles Games'

for ind in range(26,28):
    l_cat[ind] = 'phone games'

for ind in range(28,32):
    l_cat[ind] = 'CD games'

for ind in range(32,37):
    l_cat[ind] = 'Card'

for ind in range(37,43):
    l_cat[ind] = 'Movie'

for ind in range(43,55):
    l_cat[ind] = 'Books'

for ind in range(55,61):
    l_cat[ind] = 'Music'

for ind in range(61,73):
    l_cat[ind] = 'Gifts'

for ind in range(73,79):
    l_cat[ind] = 'Soft'


item_category['cats'] = l_cat
item_category.head(15)
```

Out[25]:

	item_category_name	item_category_id	cats
0	PC - Гарнитуры/Наушники	0	PC - Гарнитуры/Наушники
1	Аксессуары - PS2	1	Access
2	Аксессуары - PS3	2	Access
3	Аксессуары - PS4	3	Access
4	Аксессуары - PSP	4	Access
5	Аксессуары - PSVita	5	Access
6	Аксессуары - XBOX 360	6	Access
7	Аксессуары - XBOX ONE	7	Access
8	Билеты (Цифра)	8	Билеты (Цифра)

	item_category_name	item_category_id	cats
9	Доставка товара	9	Доставка товара
10	Игровые консоли - PS2	10	Consoles
11	Игровые консоли - PS3	11	Consoles
12	Игровые консоли - PS4	12	Consoles
13	Игровые консоли - PSP	13	Consoles
14	Игровые консоли - PSVita	14	Consoles

날짜 객체로 변환

In [26]:

```
train['date'] = pd.to_datetime(train.date,format="%d.%m.%Y")
train.head()
```

Out[26]:

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	2013-01-02	0	59	22154	6.907755	1.0
1	2013-01-03	0	25	2552	6.802395	1.0
2	2013-01-05	0	25	2552	6.802395	-1.0
3	2013-01-06	0	25	2554	7.444278	1.0
4	2013-01-15	0	25	2555	7.003065	1.0

In [27]:

```
## Pivot by monht to wide format
# 행 : shop_id, item_id
# 열 : data_block_num
# 값 : 일별 판매된 제품수(shop_id, item_id, date_block_num 교차), 결측치는 0으로
p_df = train.pivot_table(index=['shop_id','item_id'],
                         columns='date_block_num',
                         values='item_cnt_day',
                         aggfunc='sum').fillna(0.0)
print(p_df.shape)
p_df.head(15)
```

```
(424123, 34)
```

Out[27]:

	date_block_num	0	1	2	3	4	5	6	7	8	9	...	24	25	26	27
shop_id	**item_id**															
0	**30**	0.0	31.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	31	0.0	11.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	32	6.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	33	3.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	35	1.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	36	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	40	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	42	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	43	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	49	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	51	2.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	57	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	59	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	61	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
	75	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

15 rows × 34 columns

In [28]:

```
## 피벗 한 내용에 대한 인덱스를 초기화
train_cleaned_df = p_df.reset_index()
train_cleaned_df
```

Out[28]:

date_block_num	shop_id	item_id	0	1	2	3	4	5	6	7	...	24	25	26	27
0	0	30	0.0	31.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	0	31	0.0	11.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0	32	6.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	0	33	3.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	0	35	1.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
424118	59	22154	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
424119	59	22155	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0
424120	59	22162	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	9.0	4.0	1.0
424121	59	22164	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	2.0	1.0	2.0
424122	59	22167	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

424123 rows × 36 columns

In [29]:

```
# 상점 ID와 아이템 ID의 문자열로 변경
train_cleaned_df['shop_id']= train_cleaned_df.shop_id.astype('str')
train_cleaned_df['item_id']= train_cleaned_df.item_id.astype('str')

print(items.head(5))
print(item_category.head(3))
```

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

   item_category_id
0                40
1                76
2                40
3                40
4                40
        item_category_name  item_category_id                   cats
0  PC - Гарнитуры/Наушники                  0  PC - Гарнитур
ы/Наушники
1        Аксессуары - PS2                  1                 Access
2        Аксессуары - PS3                  2                 Access
```

In [30]:

```
# how{ ‘left’ , ‘right’ , ‘outer’ , ‘inner’ }, default ‘inner’
# inner : 공통된 값을 넣지 않으면 생략
# on : 키가 되는 필드
item_to_cat_df = items.merge(item_category[['item_category_id','cats']],
                             how="inner", on="item_category_id")[['item_id','cats']]
item_to_cat_df.head()
```

Out[30]:

	item_id	cats
0	0	Movie
1	2	Movie
2	3	Movie
3	4	Movie
4	5	Movie

In [31]:

```
train_cleaned_df.shape
```

Out[31]:

```
(424123, 36)
```

In [32]:

```
item_to_cat_df[['item_id']] = item_to_cat_df.item_id.astype('str')

train_cleaned_df = train_cleaned_df.merge(item_to_cat_df, how="inner", on="item_id")
print(train_cleaned_df.head() )

# Encode Categories
from sklearn import preprocessing

number = preprocessing.LabelEncoder()
train_cleaned_df[['cats']] = number.fit_transform(train_cleaned_df.cats)
train_cleaned_df = train_cleaned_df[['shop_id', 'item_id', 'cats'] + list(range(34))]
train_cleaned_df.head()
```

```
   shop_id item_id    0     1    2    3    4    5    6    7  ...   25   26  ₩
0        0      30  0.0  31.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
1        1      30  0.0  10.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
2        2      30  0.0   0.0  1.0  0.0  0.0  1.0  0.0  0.0  ...  0.0  0.0
3        3      30  0.0   4.0  5.0  2.0  2.0  1.0  0.0  0.0  ...  0.0  0.0
4        4      30  0.0   7.0  3.0  0.0  0.0  0.0  0.0  1.0  ...  0.0  0.0

    27   28   29   30   31   32   33   cats
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  Movie
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  Movie
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  Movie
3  0.0  0.0  0.0  1.0  0.0  0.0  0.0  Movie
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  Movie

[5 rows x 37 columns]
```

Out[32]:

	shop_id	item_id	cats	0	1	2	3	4	5	6	...	24	25	26	27	28	29	30
0	0	30	7	0.0	31.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1	30	7	0.0	10.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2	30	7	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	3	30	7	0.0	4.0	5.0	2.0	2.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	4	30	7	0.0	7.0	3.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 37 columns

모델 만들기

- xgboost 모델

In [34]:

```
import xgboost as xgb
param = {'max_depth':10,
         'subsample':1,
         'min_child_weight':0.5,
         'eta':0.3,
         'num_round':1000,
         'seed':1,
         'silent':0,
         'eval_metric':'rmse'}
```

xgb.DMatrix함수를 이용하여 데이터프레임을 xgBoost 행렬로 변환

In [35]:

```
progress = dict()

xgbtrain = xgb.DMatrix(train_cleaned_df.iloc[:, (train_cleaned_df.columns != 33)].values,
                       train_cleaned_df.iloc[:, train_cleaned_df.columns == 33].values)

watchlist  = [(xgbtrain,'train-rmse')]
```

In [36]:

```
%%time

bst = xgb.train(param, xgbtrain)
preds = bst.predict(xgb.DMatrix(train_cleaned_df.iloc[:,  (train_cleaned_df.columns != 33)].values)
preds
```

```
[02:17:23] WARNING: ../src/learner.cc:516:
Parameters: { num_round, silent } might not be used.

  This may not be accurate due to some parameters are only used in language bindings
but
  passed down to XGBoost core.  Or some parameters are not used but slip through thi
s
  verification. Please open an issue if you find above cases.


CPU times: user 26.3 s, sys: 1.37 s, total: 27.7 s
Wall time: 9.83 s
```

Out[36]:

```
array([0.08431637, 0.08431637, 0.055103  , ..., 0.05482858, 0.12050536,
       0.04214162], dtype=float32)
```

In [37]:

```
from sklearn.metrics import mean_squared_error

rmse = np.sqrt(mean_squared_error(preds,
                                  train_cleaned_df.iloc[:, train_cleaned_df.columns == 33].values))
print(rmse)
```

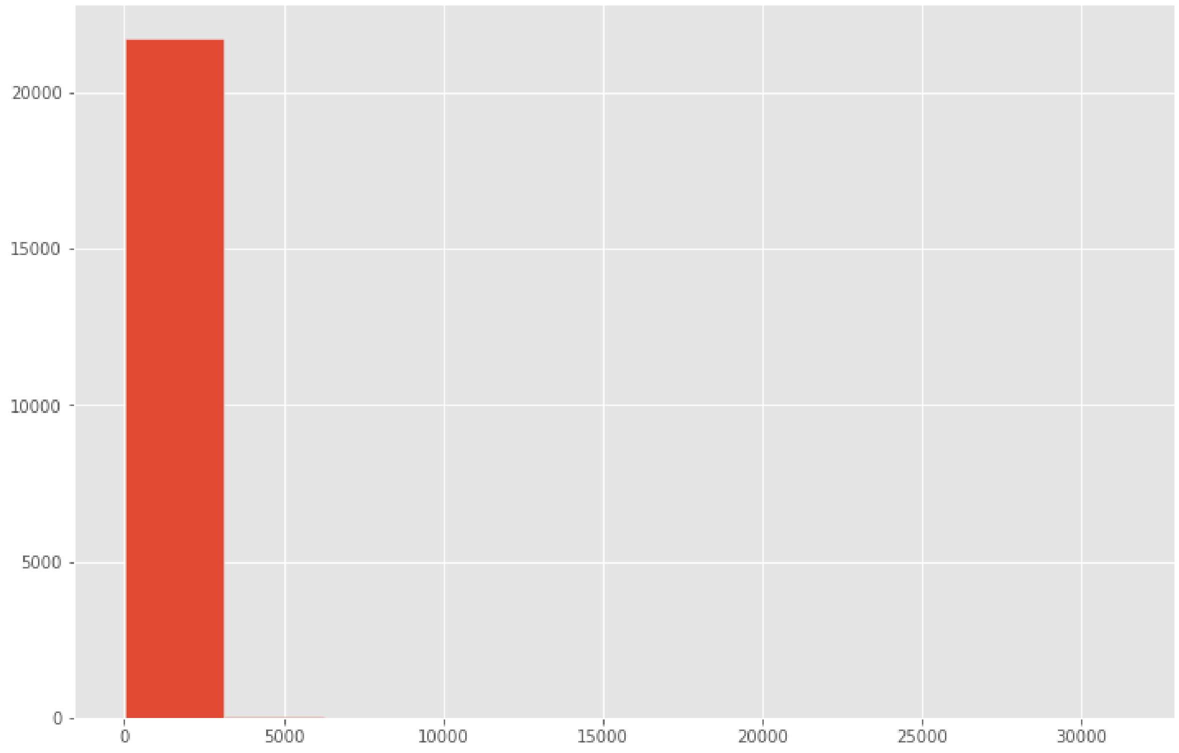
```
1.2689500930744357
```

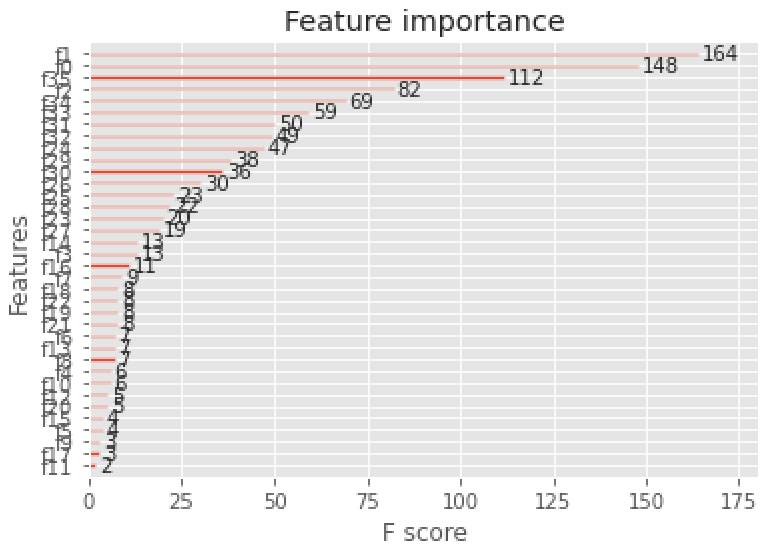
In [44]:

```
fig, ax = plt.subplots(figsize=(12,8))

# count_price.hist(figsize=(12,8))
xgb.plot_importance(bst, ax=ax)
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5cedfa5250>
```





In [39]:

```
apply_df = test
apply_df['shop_id']= apply_df.shop_id.astype('str')
apply_df['item_id']= apply_df.item_id.astype('str')

apply_df = test.merge(train_cleaned_df, how = "left", on = ["shop_id", "item_id"]).fillna(0.0)
apply_df.head()
```

Out[39]:

	ID	shop_id	item_id	cats	0	1	2	3	4	5	...	24	25	26	27	28	29	30
0	0	5	5037	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.0	0.0	0.0	0.0	1.0	1.0	1.0
1	1	5	5320	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2	5	5233	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	2.0	0.0
3	3	5	5232	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	4	5	5268	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 38 columns

In [ ]:

```
# Move to one month front
d = dict(zip(apply_df.columns[4:],list(np.array(list(apply_df.columns[4:])) - 1)))

apply_df  = apply_df.rename(d, axis = 1)
```

In [ ]:

```
preds = bst.predict(xgb.DMatrix(apply_df.iloc[:, (apply_df.columns != 'ID') & (apply_df.columns !=
preds
```

In [ ]:

```
# Normalize prediction to [0-20]
preds = list(map(lambda x: min(20,max(x,0)), list(preds)))
sub_df = pd.DataFrame({'ID':apply_df.ID,'item_cnt_month': preds })
sub_df.describe()
```

In [ ]:

```
sub_df.to_csv('Submission_PredictSales.csv',index=False)
```

In [ ]:

```

```