상점의 매출 미래 예측 (한달 기준)

- 대회 URL : 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

In [89]:
```
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
```

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 [90]:
```
train = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales
test = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales/
items = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales
item_category = pd.read_csv('/kaggle/input/competitive-data-science-predict-futu
shops = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales
sub = pd.read_csv("/kaggle/input/competitive-data-science-predict-future-sales/s
```

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)

구분	컬럼명	설명	값
shops	shop_id	상점 고유 ID	0~59

In [91]:
```
# 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')
```

In [92]:
```
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 [93]:
```
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 [94]:
```
# data: 중복을 제거할 데이터프레임
# subset: 중복을 확인할 열(들)의 리스트
# before = data.shape[0] : 중복 제거 전의 데이터프레임의 행 수를 before 변수에 저
# drop_duplicates 메서드를 사용하여 지정된 subset 열을 기준으로 중복된 행을 제거합
#   keep='first'는 첫 번째 중복된 행을 유지하고 나머지를 제거
# data.reset_index(drop=True, inplace=True)  중복 제거 후 인덱스를 재설정합니다.
#   drop=True는 기존 인덱스를 삭제하고 새로운 인덱스를 생성
# after = data.shape[0] : 중복 제거 후의 데이터프레임의 행 수를 after 변수에 저장
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)  # 인덱스 재설정
    print('(처리후) 데이터 행열:', data.shape)
```

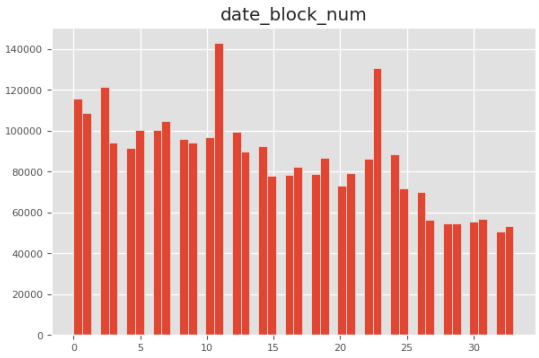
```
    after = data.shape[0]
    print('Total Duplicate:', before-after)
```

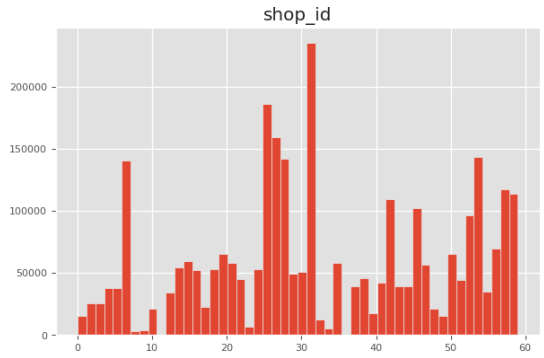
In [95]:
```
eda(train)
```

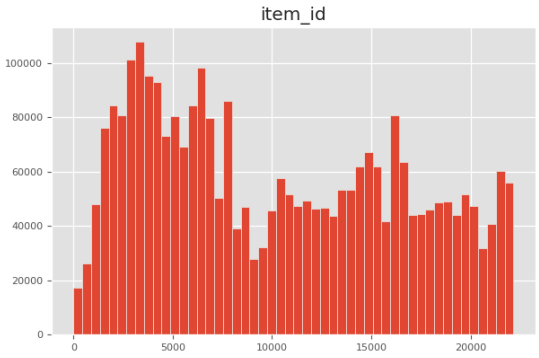
```
----------Top-5----------
         date  date_block_num  shop_id  item_id  item_price  item_cnt_day
0  02.01.2013               0       59    22154      999.00           1.0
1  03.01.2013               0       25     2552      899.00           1.0
2  05.01.2013               0       25     2552      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)
```

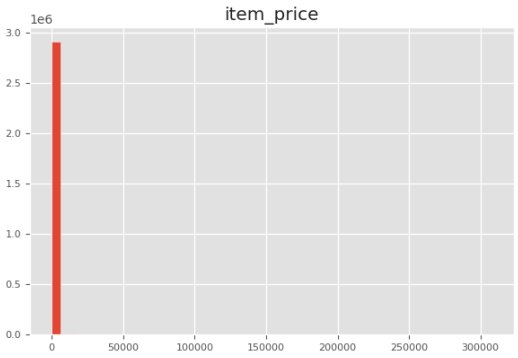
In [96]:
```
graph_insight(train)
```

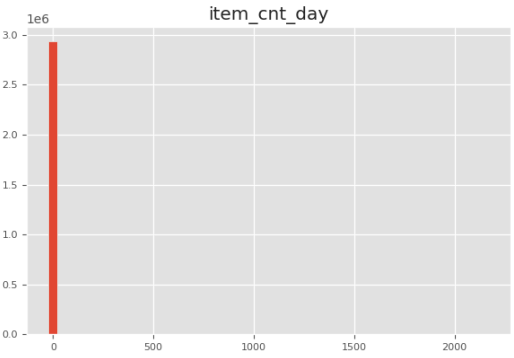
```
{dtype('O'), dtype('int64'), dtype('float64')}
```





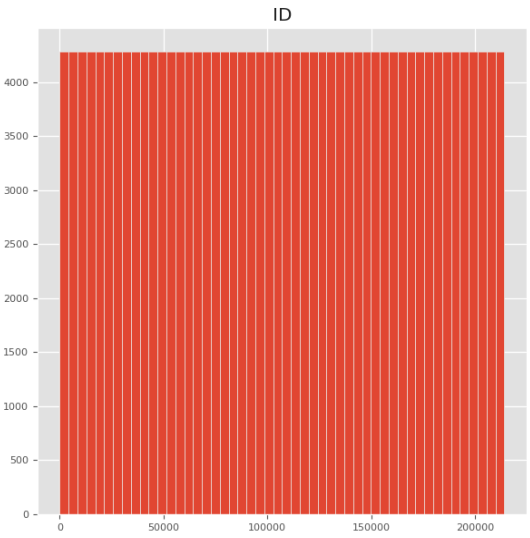


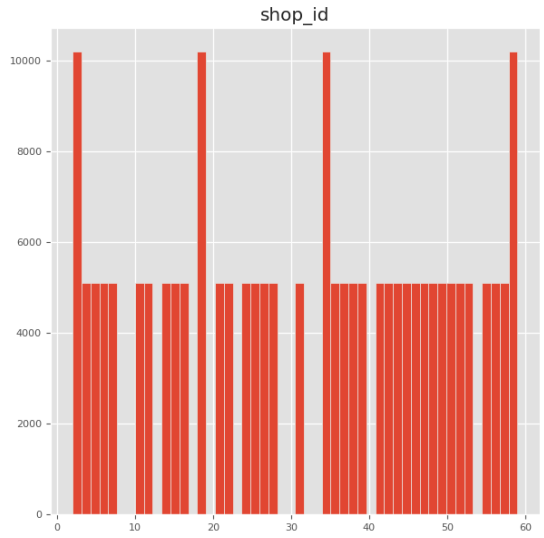


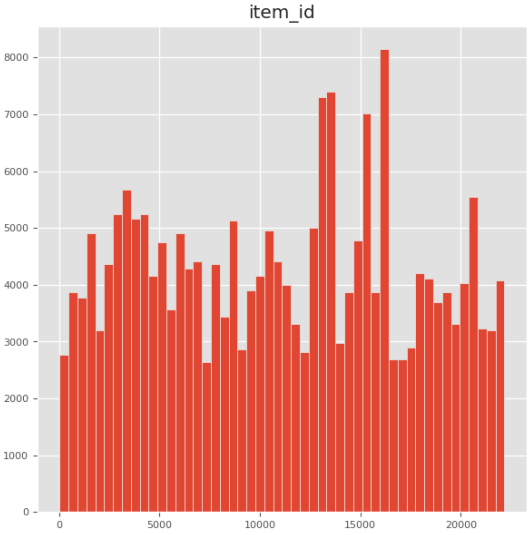


In [97]:
```
graph_insight(test)
```

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





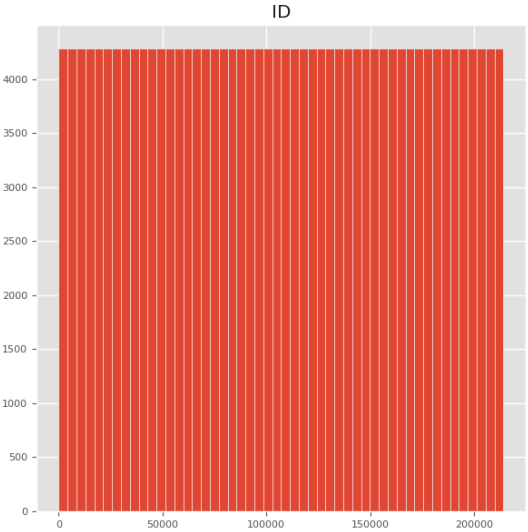


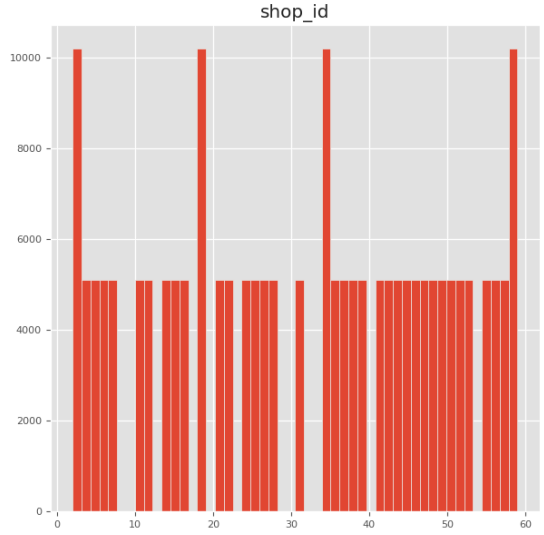
In [98]:
```
### 중복 데이터 제거
subset = ['date', 'date_block_num', 'shop_id', 'item_id','item_cnt_day']
drop_duplicate(train, subset = subset)
```

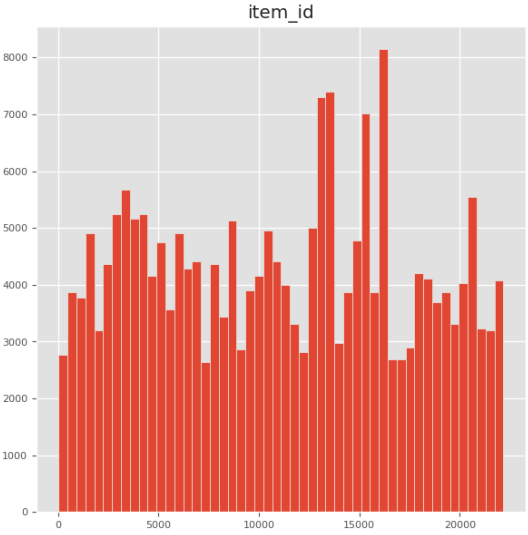
```
(처리전) 데이터 행열: (2935849, 6)
(처리후) 데이터 행열: (2935825, 6)
Total Duplicate: 24
```

In [99]:
```
# 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')}
```





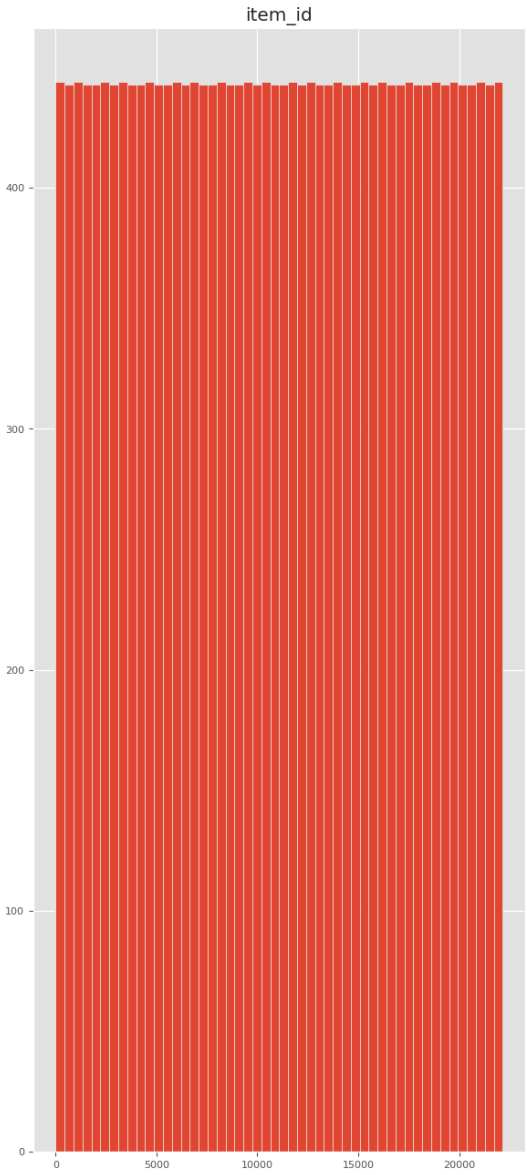


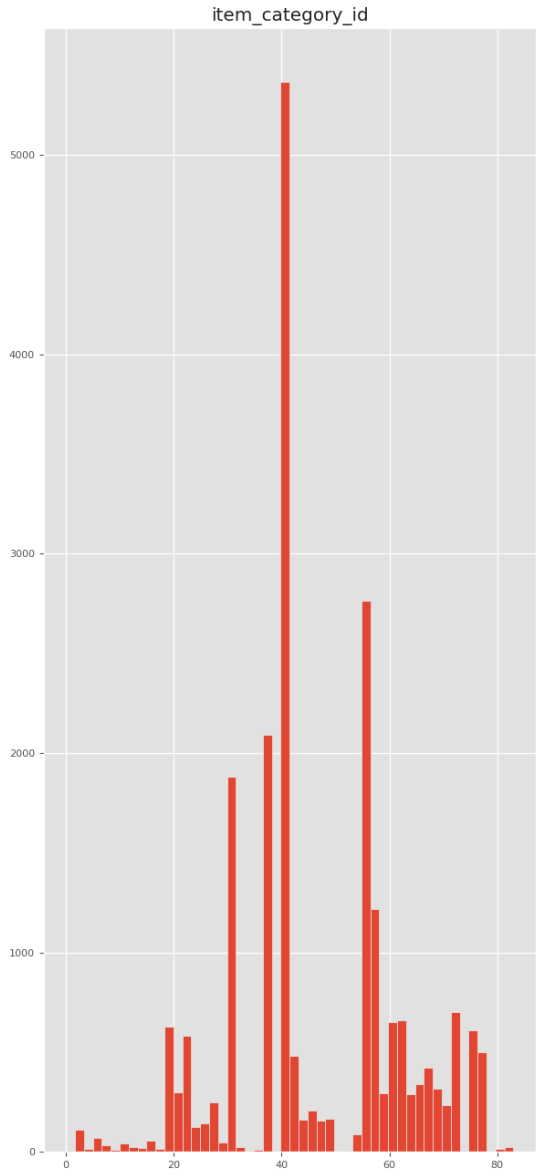
item 데이터

In [100…

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





item category

In [101…

```
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)
```

shop 데이터

In [102…

```
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)
```

데이터의 최대, 최소 및 평균, 중앙값

In [103…

```
### 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())
```

아웃라이어

- item_price에 0이하와 300000이상은 제외

In [104…

```
# -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 [105…

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

Out[105...

date_block_num	date	shop_id	item_id	
0	02.01.201303.01.201305.01.201306.01.201315.01....	3416958	1183925474	8
1	21.02.201314.02.201321.02.201313.02.201324.02....	3111541	1076016145	1
2	03.03.201306.03.201302.03.201317.03.201302.03....	4016391	1220887356	8
3	16.04.201327.04.201329.04.201328.04.201306.04....	3164924	971331915	0
4	10.05.201317.05.201304.05.201303.05.201322.05....	3093967	950370015	5

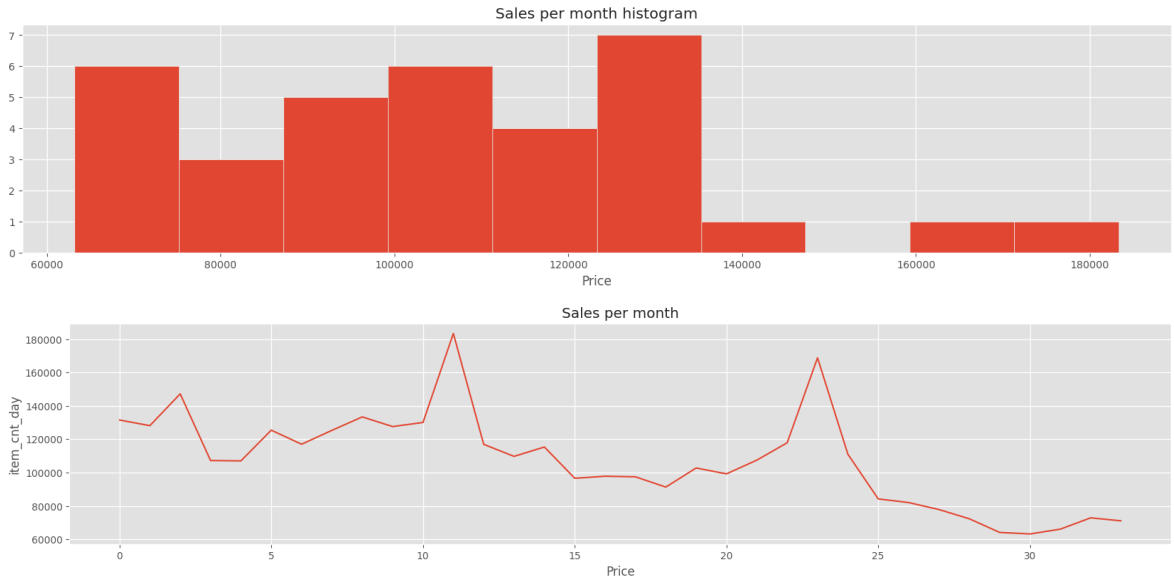
In [106...

```
# 월별 매출
# train 데이터프레임을 date_block_num(월을 나타내는 열)으로 그룹화하고,
# 각 그룹의 item_cnt_day(일일 판매 수량)의 합계. 이 결과는 각 월별 총 판매 수량
train.groupby('date_block_num').sum()['item_cnt_day'].hist(figsize = (20,4))
plt.title('Sales per month histogram')
plt.xlabel('Price')

# Seaborn의 lineplot 함수를 사용하여,
# 월별 총 판매 수량의 선 그래프를 생성합니다. 이 그래프는 시간에 따른 판매 추세
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')
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[106...

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



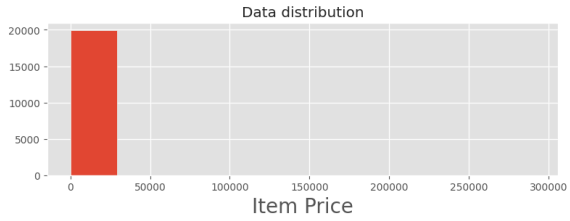
데이터 분포 확인

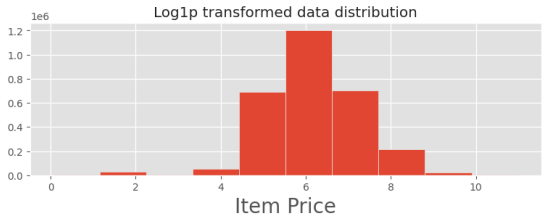
In [107…
```
train.item_price.value_counts().sort_index(ascending=False)
```

Out[107…
```
item_price
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: count, Length: 19991, dtype: int64
```

In [108…
```
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('Data distribution')  # 데이터 분포

plt.subplot(222)
train.item_price.map(np.log1p).hist(figsize=(20,6))
plt.xlabel('Item Price', fontsize=20);
plt.title('Log1p transformed data distribution') # log1p 변환 데이터 분포
train.loc[:,'item_price'] = train.item_price.map(np.log1p)
```

```
Min Value: 0.07
Max Value: 59200.0
Average Value: 890.7514892291624
Center Point of Data: 399.0
```





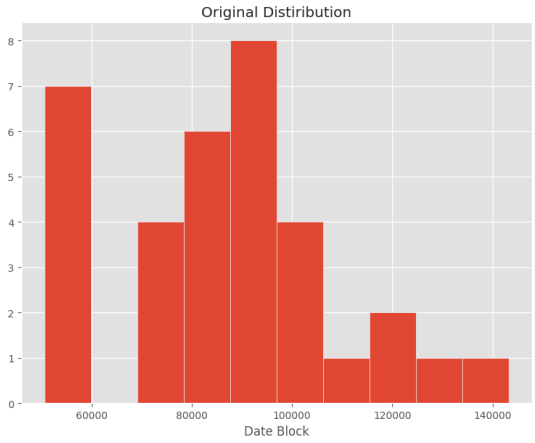
In [109…
```
print( train.date_block_num.unique() )
print( train.shop_id.unique() )
print( train.item_id.unique() )
```

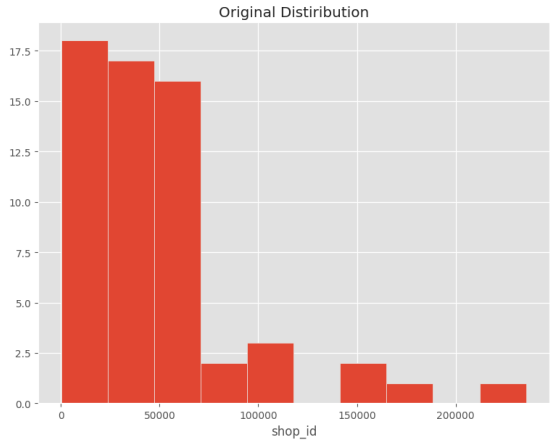
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33]
[59 25 24 23 19 22 18 21 28 27 29 26  4  6  2  3  7  0  1 16 15  8 10 14
 13 12 53 31 30 32 35 56 54 47 50 42 43 52 51 41 38 44 37 46 45  5 57 58
 55 17  9 49 39 40 48 34 33 20 11 36]
[22154 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

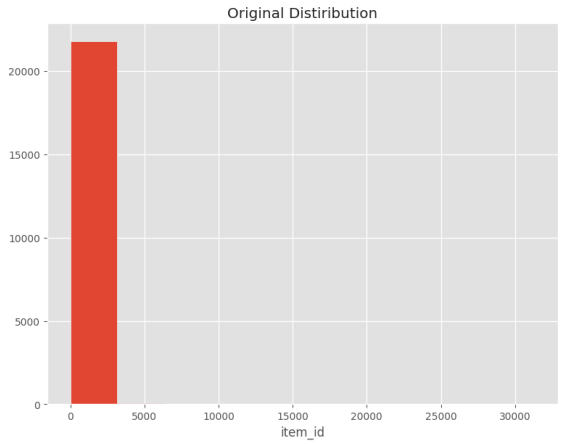
In [110…
```
# unresanable_data(train['date_block_num'])

### 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[110… Text(0.5, 1.0, 'Original Distiribution')







In [111…
```
list(item_category.item_category_name)
```

Out[111…

```
['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 - МАС (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 이름을 영어로 변경

In [112…

```
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[112...

	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)
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 [113...

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

Out[113...

	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 [114...

```
## 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
```

```
# 피벗 테이블 형식으로 데이터를 변경.
# index=['shop_id', 'item_id'] : 피벗 테이블의 행 인덱스를 shop_id와 item_id로 설
# columns='date_block_num' : 피벗 테이블의 열을 date_block_num으로 설정
# values='item_cnt_day' : 피벗 테이블의 값으로 item_cnt_day를 설정
# aggfunc='sum' : 동일한 shop_id, item_id, date_block_num 조합에 대해 여러 개의
# .fillna(0.0) : 피벗 테이블 생성 후 결측치(NaN)를 0.0으로 한다.
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[114…

	date_block_num	0	1	2	3	4	5	6	7	8	9	…	24	25	2
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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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

15 rows × 34 columns

In [115…

```
## 피벗 한 내용에 대한 인덱스를 초기화
# train_cleaned_df = p_df.reset_index() : reset_index() 메서드를 사용하여 피벗 테
#    이 메서드는 기존 인덱스를 기본 정수 인덱스로 변경하고, 기존 인덱스를 새로운
train_cleaned_df = p_df.reset_index()
train_cleaned_df
```

Out[115…

	date_block_num	shop_id	item_id	0	1	2	3	4	5	6	7	…	24	25	2
0	0	30	0.0	31.0	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
2	0	32	6.0	10.0	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
4	0	35	1.0	14.0	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
424119	59	22155	0.0	0.0	0.0	0.0	0.0	0.0	1.0	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
424121	59	22164	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	2.0	1
424122	59	22167	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 [116…

```
# 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')

train_cleaned_df
```

Out[116…

	date_block_num	shop_id	item_id	0	1	2	3	4	5	6	7	…	24	25	2
0	0	30	0.0	31.0	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
2	0	32	6.0	10.0	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
4	0	35	1.0	14.0	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
424119	59	22155	0.0	0.0	0.0	0.0	0.0	0.0	1.0	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
424121	59	22164	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	2.0	1
424122	59	22167	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 [117…

```
items.head(3)
```

Out[117...

	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

In [118...

```
item_category.head(3)
```

Out[118...

	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

items와 item_category를 병합

In [124...

```
# items 데이터프레임과 item_category 데이터프레임을 item_category_id를 기준으로 L
# item_id와 cats 열만 포함된 새로운 데이터프레임 item_to_cat_df를 생성

# how{'left', 'right', 'outer', 'inner'}, default 'inner'
# how="inner": 병합 방법을 지정합니다. inner는 두 데이터프레임에서 공통된 키 값이
# 즉, item_category_id가 두 데이터프레임 모두에 존재하는 경우에만 결과에 포함

# 키 필드 (on):
# on="item_category_id": 병합할 때 사용할 키 필드를 지정합니다.
# 여기서는 item_category_id가 두 데이터프레임에서 공통적으로 존재하는 열.

# 병합 수행:
# items.merge(item_category[['item_category_id','cats']], how="inner", on="item_
# items 데이터프레임과 item_category 데이터프레임의 item_category_id와 cats 열만

# 결과 열 선택
# [['item_id','cats']]: 병합 결과에서 item_id와 cats 열만 선택하여 최종 데이터프
item_to_cat_df = items.merge(item_category[['item_category_id','cats']],
                             how="inner",
                             on="item_category_id")[['item_id','cats']]

print( item_to_cat_df.shape )
item_to_cat_df.head()
```

```
(22170, 2)
```

Out[124...

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

In [125…
```
print( train_cleaned_df.shape )
print( item_to_cat_df.shape )
```

```
(424123, 38)
(22170, 2)
```

아이템-카테고리 데이터를 처리하고, 카테고리 정보를 인코딩하여 최종 데이터프레임을 생성하는 작업을 수행

- item_id 열의 데이터 타입을 문자열로 변환
- train_cleaned_df 데이터프레임과 카테고리 정보가 포함된 item_to_cat_df를 병합
- 카테고리 정보를 레이블 인코딩하여 숫자 형태로 변환
- 최종 데이터프레임을 특정 열 순서로 재구성

In [126…
```
train_cleaned_df
```

Out[126…

	shop_id	item_id	0	1	2	3	4	5	6	7	…	26	27	28	29
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	…	4.0	1.0	1.0	0.0
424121	59	22164	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	1.0	2.0	0.0	0.0
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 × 38 columns

In [127…
```
item_to_cat_df['item_id'] = item_to_cat_df['item_id'].astype('str')

# merge() 메서드를 사용하여 train_cleaned_df와 item_to_cat_df를 병합
# how="inner": 내부 조인(inner join)을 수행합니다. 이는 두 데이터프레임에서 공통된
# 즉, train_cleaned_df와 item_to_cat_df 모두에 존재하는 item_id에 대해서만 결과가
train_cleaned_df = train_cleaned_df.merge(item_to_cat_df, how="inner", on="item_

print( train_cleaned_df.shape )
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(

print( train_cleaned_df.shape )
train_cleaned_df.head()
```

```
(424123, 39)
(424123, 37)
```

Out[127…

	shop_id	item_id	cats	0	1	2	3	4	5	6	…	24	25	26	27	28
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
1	0	31	7	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	7	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	7	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	7	1.0	14.0	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 구축 및 학습

In [128…

```
import xgboost as xgb
param = {'max_depth':10,     # 트리의 최대 깊이를 설정. 값이 3~10 사이에서 성능이
         'subsample':1,      # 각 트리를 훈련하는 데 사용할 데이터 샘플의 비율( 1
         'min_child_weight':0.5, # 자식 노드의 최소 가중치 합을 설정. 이 값이 크
         'eta':0.3,   #  (Learning Rate) : 각 트리가 기여하는 단계 크기로, 학습률
         'num_round':1000,   # 부스팅 반복 횟수를 설정합니다. 즉, 트리를 몇 개 만
         'seed':1,    # 난수 생성 시드로, 결과의 재현성을 보장
         'silent':0,  # 학습 과정에서 출력되는 로그의 양을 결정. 0일 경우 모든 로
         'eval_metric':'rmse'} # 모델의 성능을 평가하기 위해 사용할 메트릭을 설정
```

- 이 코드는 XGBoost 모델을 훈련하기 위해 데이터를 준비하고, 훈련 과정을 모니터링하기 위한 watchlist를 설정하는 부분

In [131…

```
train_cleaned_df
```

Out[131...

	shop_id	item_id	cats	0	1	2	3	4	5	6	...	24	25	26	27
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
1	0	31	7	0.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0	32	7	6.0	10.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	0	33	7	3.0	3.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	0	35	7	1.0	14.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
424118	59	22154	7	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
424119	59	22155	7	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0
424120	59	22162	7	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	7	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	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

424123 rows × 37 columns

In [129...

```
progress = dict()

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

# "watchlist"는 모델 훈련 중에 특정 데이터셋의 성능을 모니터링하기 위해 사용
watchlist  = [(xgbtrain,'train-rmse')]
```

In [144...

```
%%time

model_xgb = xgb.train(param, xgbtrain)
preds = model_xgb.predict(xgb.DMatrix(train_cleaned_df.iloc[:,  (train_cleaned_d
preds
```

```
/opt/conda/lib/python3.10/site-packages/xgboost/core.py:160: UserWarning: [13:37:
47] WARNING: /workspace/src/learner.cc:742:
Parameters: { "num_round", "silent" } are not used.

  warnings.warn(smsg, UserWarning)
CPU times: user 4.83 s, sys: 569 ms, total: 5.4 s
Wall time: 3.23 s
```

Out[144...

```
array([0.03533427, 0.03533427, 0.00756854, ..., 0.16320726, 0.17273381,
       0.083051  ], dtype=float32)
```

In [145...

```
from sklearn.metrics import mean_squared_error

rmse = np.sqrt(mean_squared_error(preds,
                                  train_cleaned_df.iloc[:, train_cleaned_df.colu
print(rmse)
```

```
1.2720128040857526
```

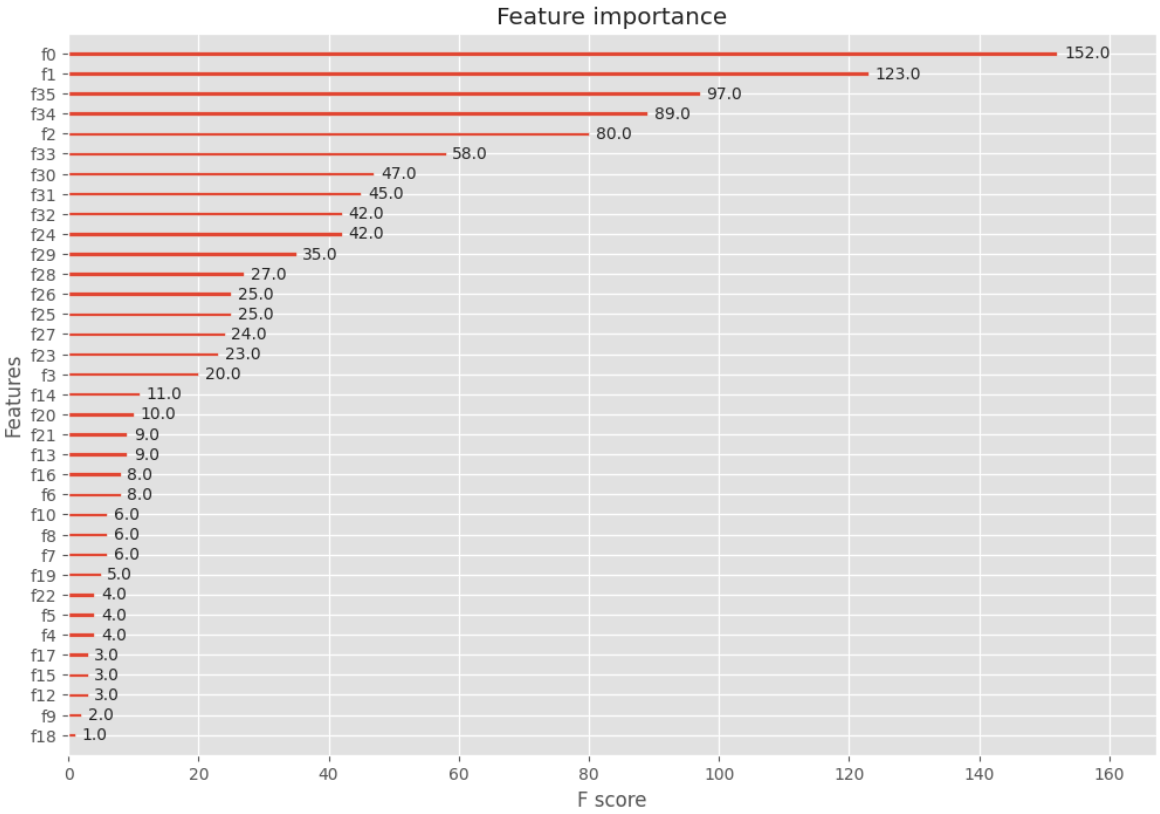
In [146...

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

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

Out[146…

```
<Axes: title={'center': 'Feature importance'}, xlabel='F score', ylabel='Features'>
```



In [147…

```
test.head()
```

Out[147…

	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 [148…

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

# 설명: test 데이터프레임과 train_cleaned_df 데이터프레임을 shop_id와 item_id를 
# how = "left"는 test 데이터프레임의 모든 행을 유지하고, train_cleaned_df에서 일
# apply_df는 test의 모든 행과 train_cleaned_df에서 일치하는 데이터를 포함
# apply_df.fillna(0.0): 병합 후 생성된 NaN 값을 0.0으로 대체
apply_df = test.merge(train_cleaned_df, how = "left", on = ["shop_id", "item_id"
apply_df.head()
```

Out[148...

	ID	shop_id	item_id	cats	0	1	2	3	4	5	...	24	25	26	27	28	29
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	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
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
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
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

5 rows × 38 columns

In [149...

```
# 데이터프레임의 열 이름을 한 달 전의 값으로 이동시키는 작업
# apply_df 데이터프레임의 열 이름을 변경하여, 특정 열의 이름을 한 달 전의 값으로
# apply_df.columns[4:]: apply_df의 열 이름 중 4번째 열부터 끝까지의 열 이름을 선
# list(np.array(list(apply_df.columns[4:])) - 1):
#    apply_df.columns[4:]를 리스트로 변환한 후, NumPy 배열로 변환
#    이 배열에서 각 열 이름에 대해 1을 빼줍니다.
#    이 배열에서 각 열 이름에 대해 1을 빼줍니다. 이는 열 이름이 숫자로 되어 있다고
#    예를 들어, 열 이름이 [34, 35, 36]이라면, 이 부분은 [33, 34, 35]로 변환.

d = dict(zip(apply_df.columns[4:],list(np.array(list(apply_df.columns[4:])) - 1)

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

Out[149...

	ID	shop_id	item_id	cats	-1	0	1	2	3	4	...	23	24	25	2
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
214195	214195	45	18454	8.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.0	1.0	1.0	0
214196	214196	45	16188	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
214197	214197	45	15757	8.0	1.0	0.0	0.0	0.0	0.0	0.0	...	1.0	1.0	0.0	0
214198	214198	45	19648	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
214199	214199	45	969	7.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0

214200 rows × 38 columns

In [151...

```
# 특정 열을 제외한 모든 열을 선택.
preds = model_xgb.predict(xgb.DMatrix(apply_df.iloc[ : ,
                                                     (apply_df.columns != 'ID') &
preds
```

Out[151…

```
array([0.29781207, 0.19222318, 0.69343615, ..., 0.04283362, 0.17335418,
       0.06511665], dtype=float32)
```

In [139…

```
# 모델의 예측 결과를 정규화하고, 이를 새로운 데이터프레임으로 구성한 후, 통계적 요
# preds 리스트의 각 요소를 0과 20 사이로 제한
#   map 함수와 lambda 함수를 사용하여 각 예측값을 처리
#    max(x, 0): 예측값 x가 0보다 작으면 0으로 설정
#    min(20, ...): 그 다음, 20보다 크면 20으로 설정합니다. 즉, 예측값이 20을 초과
#    결과적으로, 이 작업은 preds의 모든 예측값을 0과 20 사이로 제한
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()
```

Out[139…

	ID	item_cnt_month
count	214200.000000	214200.000000
mean	107099.500000	0.279595
std	61834.358168	0.679682
min	0.000000	0.000000
25%	53549.750000	0.155149
50%	107099.500000	0.183499
75%	160649.250000	0.228294
max	214199.000000	20.000000

In [152…

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