KYUNG HEE UNIVERSITY

KHU DATA

목차

1 서론

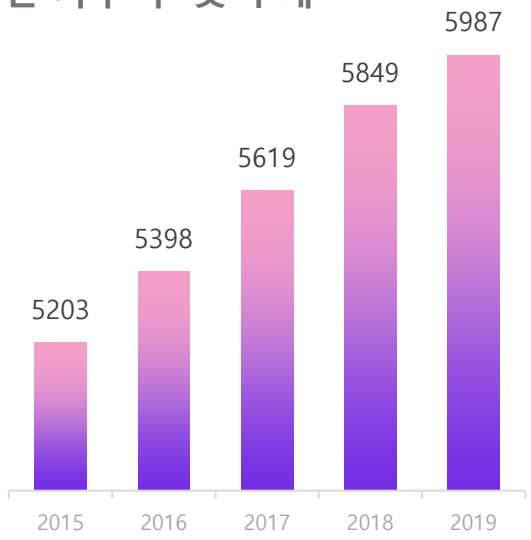
유통 트렌드의 변화

주문형 서비스(On-Demand)의 증가

모바일, 온라인 플랫폼을 통한 유통구조 단순화

방송편성의 차별화 필요성 대두!

연도별 1인 가구 수 및 추계



출처: 보건복지부

소비 트렌드의 변화

1인 가구 증가

모바일 채널 매출 증가

연령대별 소비 구조의 변화

2

목표 설

상품 매출 실적 향상

다양한 데이터를 활용하여
상품 매출 실적 향상 기여

MAPE (Mean Absolute Percentage Error)

$$MAPE = \frac{100}{n}\sum\frac{|Prediction - Actual|}{Actual}$$

평균 절대 오차 비율 오차(MAPE) 감소

평균 절대 오차 비율 오차 (MAPE) 감소
상품별 예측 학습 후 예측 성능이 향상된 모델 제시

3 데이터 전처

Data Type 확인

Train Data Column **7개**

Test Data Column **7개**

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38309 entries, 0 to 38308
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   방송일시    38309 non-null  datetime64[ns]
 1   노출(분)   21525 non-null  float64
 2   마더코드    38309 non-null  int64
 3   상품코드    38309 non-null  int64
 4   상품명     38309 non-null  object
 5   상품군     38309 non-null  object
 6   판매단가    38309 non-null  int64
 7   취급액     37372 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(3), object(2)
memory usage: 2.3+ MB
```

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2891 entries, 0 to 2890
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   방송일시    2891 non-null   datetime64[ns]
 1   노출(분)   1780 non-null   float64
 2   마더코드    2891 non-null   int64
 3   상품코드    2891 non-null   int64
 4   상품명     2891 non-null   object
 5   상품군     2891 non-null   object
 6   판매단가    2891 non-null   int64
 7   취급액     0 non-null      float64
dtypes: datetime64[ns](1), float64(2), int64(3), object(2)
memory usage: 180.8+ KB
```

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

```
방송일시        0
노출(분)    16784
마더코드        0
상품코드        0
상품명         0
상품군         0
판매단가        0
취급액       937
dtype: int64
```

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

```
방송일시        0
노출(분)     1111
마더코드        0
상품코드        0
상품명         0
상품군         0
판매단가        0
취급액       2891
dtype: int64
```

결측치 확인

Column	Train Data	Test Data
노출(분)	16,784개	1,111개
취급액	937개	2,891개

3 데이터 전처

Train Data, Test Data의 '노출(분)' Column 결측치 채우기

fillna 함수 사용

```
train['노출(분)'] = train['노출(분)'].fillna(method='ffill')
test['노출(분)'] = test['노출(분)'].fillna(method='ffill')
```

```
print('무형 상품 갯수')
print('train:',len(train[train['상품군']=='무형']))
print('test:',len(test[test['상품군']=='무형']))
```

```
무형 상품 갯수
train: 937
test: 175
```

Train Data, Test Data의 '상품군' Column이 '무형'인 Data 제거

```
train = train[train['상품군']!='무형']
test = test[test['상품군']!='무형']
```

Column	Train Data	Test Data
상품군	937개	175개

3 데이터 전처

Train Data, Test Data에 '성별' Column 추가

'상품명'에 있는 키워드로 구분

남자 키워드 有	여자 키워드 有	모두 X
1	2	3

```python
def find_gender(name):
    if '남성' in name:
        return 1
    elif '여성' in name:
        return 2
    else:
        return 3

# 성별 칼럼 생성
train['성별'] = train['상품명'].map(find_gender)
test['성별'] = test['상품명'].map(find_gender)

train['성별'].value_counts()
```

```
3    33101
1     2583
2     1688
Name: 성별, dtype: int64
```

```python
test['성별'].value_counts()
```

```
3    2506
1     153
2      57
Name: 성별, dtype: int64
```

```python
def find_payment(name):
    if '무이자' in name or '(무)' in name:
        return 1
    elif '일시불' in name or '(일)' in name:
        return 2
    else:
        return 3

# 지불 방법 칼럼생성
train['지불'] = train['상품명'].map(find_payment)
test['지불'] = test['상품명'].map(find_payment)

train['지불'].value_counts()
```

```
3    23106
1     7146
2     7120
Name: 지불, dtype: int64
```

```python
test['지불'].value_counts()
```

```
3    2260
1     228
2     228
Name: 지불, dtype: int64
```

Train Data, Test Data에 '지불' Column 추가

'상품명'에 있는 키워드로 구분

무이자 키워드 有	일시불 키워드 有	모두 X
1	2	3

3 데이터 전처

```python
def get_progress(dataset):

    dataset = dataset.sort_values(['상품명','방송일시'])
    dataset.reset_index(drop=True, inplace=True)
    #초깃값 설정
    times = [1] # 첫 타임부터 1,2,3,...
    cur_date = dataset.loc[0,'방송일시'].date().strftime('%Y-%m-%d')
    cur_time = 1

    for idx in range(1,len(dataset)):
        last_br = dataset.loc[idx-1,'방송일시'].to_pydatetime()
        during = datetime.timedelta(minutes = int(dataset.loc[idx-1,'노출(분)']))
        cur_br = dataset.loc[idx,'방송일시'].to_pydatetime()

        if last_br + during == cur_br:
            cur_time += 1
            times.append(cur_time)

        else:
            cur_time = 1
            times.append(cur_time)

    dataset['방송진행도'] = times
    dataset = dataset.sort_values(['방송일시','상품명'])
    dataset.reset_index(drop=True, inplace=True)

    # 방송별로 묶기
    same_br = [1,1]
    cur_count = 1

    for idx in range(2,len(dataset)):
        cur_time = dataset.loc[idx,'방송진행도']
        if cur_time == 1 and cur_time != dataset.loc[idx-1,'방송진행도'] :
            cur_count+=1
        elif cur_time == 1 and dataset.loc[idx-1,'마더코드'] != dataset.loc[idx,'마더코드']:
            cur_count+=1
        same_br.append(cur_count)

    dataset['방송레이블'] = same_br

    # 방송별로 0과 1사이 값으로 스케일링
    for label in set(dataset['방송레이블']):
        max_val = dataset.loc[dataset['방송레이블']==label,'방송진행도'].max()
        dataset.loc[dataset['방송레이블']==label,'방송진행도'] /= max_val

    del dataset['방송레이블']

    return dataset
```

Train Data, Test Data에 '방송진행도' Column 추가

동일한 상품을 연달아 방송하는 경우 有

방송 초반부보다는 중, 후반부에 상품이 더 잘 팔린다는 가정

Example

	방송일시	노출(분)	마더코드	상품코드	상품명	상품군	판매단가	취급액	성별	지불	방송진행도
0	2019-01-01 06:00:00	20.0	100346	201072	테이트 남성 셀린니트3종	의류	39900	2099000.0	1	3	0.333333
1	2019-01-01 06:00:00	20.0	100346	201079	테이트 여성 셀린니트3종	의류	39900	4371000.0	2	3	0.333333
2	2019-01-01 06:20:00	20.0	100346	201072	테이트 남성 셀린니트3종	의류	39900	3262000.0	1	3	0.666667
3	2019-01-01 06:20:00	20.0	100346	201079	테이트 여성 셀린니트3종	의류	39900	6955000.0	2	3	0.666667
4	2019-01-01 06:40:00	20.0	100346	201072	테이트 남성 셀린니트3종	의류	39900	6672000.0	1	3	1.000000

	AM 6:00	AM 6:20	AM 6:40
방송진행도	0.33	0.67	1.00

3 데이터 전처

```python
def date_to_day(date):
    day = str(date.weekday())
    if day == '0':
        day = '월요일'
    elif day == '1':
        day = '화요일'
    elif day == '2':
        day = '수요일'
    elif day == '3':
        day = '목요일'
    elif day == '4':
        day = '금요일'
    elif day == '5':
        day = '토요일'
    else:
        day = '일요일'
    return day
```

```python
train['요일'] = train['방송일시'].apply(lambda x: date_to_day(x))
```

```python
test['요일'] = test['방송일시'].apply(lambda x: date_to_day(x))
```

```python
train['요일(숫자)'] = train['방송일시'].apply(lambda x: int(x.weekday()))
```

```python
test['요일(숫자)'] = test['방송일시'].apply(lambda x: int(x.weekday()))
```

Train Data, Test Data에 '요일' Column 추가

요일에 따른 취급액의 차이가 있다고 판단하여 추가

	월요일	화요일	수요일
결과	0	1	2

	목요일	금요일	토요일	일요일
결과	3	4	5	6

3 데이터 전처

```
def get_info(keyword):
    """
    Args:
        keyword(string): 검색할 키워드
    Returns:
        url(string): keyword가 포함된 url
    """
    url = "https://search.shopping.naver.com/search/all?query=" + keyword +"&cat_id=&frm=NVSHATC"

    req = requests.get(url)
    cont = req.content
    soup = BeautifulSoup(cont, 'html.parser')

    # [shop_name, numReview, score, shop_price]

    # 제품광고가 있는 경우
    try:
        shop_name = soup.select_one("div.style_content__2T20F > ul > div > div:nth-of-type(1) > li > div > div.basicList_info_a
        # 리뷰 있는 경우
        try:
            numReview = soup.select_one("#__next > div > div.container > div > div.style_content_wrap__1PzEo > div.style_conten
            numReview = int(numReview.replace(',', ''))
        # 리뷰 없는 경우
        except:
            numReview = "NULL"
        try:
            score = soup.select_one("#__next > div > div.container > div > div.style_content_wrap__1PzEo > div.style_content__2
            score = float(score.replace('별점 ', ''))
        except:
            score = "NULL"
        shop_price = soup.select_one("#__next > div > div.container > div > div.style_content_wrap__1PzEo > div.style_content__
        shop_price = int(shop_price.replace('원', '').replace(',', ''))


    # 제품광고가 없는 경우
    except:
        idx = keyword.rfind(' ')
        new_keyword = keyword[:idx]
        url = "https://search.shopping.naver.com/search/all?query=" + new_keyword +"&cat_id=&frm=NVSHATC"

        req = requests.get(url)
        cont = req.content
        soup = BeautifulSoup(cont, 'html.parser')

        try:
            shop_name = soup.select_one("div.style_content_wrap__1PzEo > div.style_content__2T20F > ul > div > div:nth-child(1)
            print(shop_name)
            try:
                numReview = soup.select_one("#__next > div > div.container > div > div.style_content_wrap__1PzEo > div.style_co
                numReview = int(numReview.replace(',', ''))
            # 리뷰 없는 경우
            except:
                numReview = "NULL"
            try:
                score = soup.select_one("#__next > div > div.container > div > div.style_content_wrap__1PzEo > div.style_conten
                score = float(score.replace('별점 ', ''))
            except:
                score = "NULL"
            shop_price = soup.select_one("#__next > div > div.container > div > div.style_content_wrap__1PzEo > div.style_conte
            shop_price = int(shop_price.replace('원', '').replace(',', ''))

        except Exception as e:
            print(e)
            print(keyword, new_keyword, '도 없음')

    print(url)
    info = dict()
    try:
        info['shop_name'] = shop_name
        info['numReview'] = numReview
        info['score'] = score
        info['shop_price'] = shop_price
    except:
        info['shop_name'] = "NULL"
        info['numReview'] = "NULL"
        info['score'] = "NULL"
        info['shop_price'] = "NULL"

    return info
```

외부변수인 네이버쇼핑에 속한 Column 4개 추가

'상품이름', '상품가격', '리뷰 개수', '평점'

```
naver_train = pd.read_excel('../data/navershop_train.xlsx')
naver_test = pd.read_excel('../data/navershop_test.xlsx')
```

```
train = pd.concat([train, naver_train], axis=1)
test = pd.concat([test, naver_test], axis=1)
```

3 데이터 전처

```
from konlpy.tag import Hannanum
hannanum = Hannanum()
```

Train Data, Test Data에 형태소 단위로 나눈 '형태소_상품명' Column 추가

'상품명' Column 분석을 명확하게 하기 위해 형태소 분석 실시

띄어쓰기가 되지 않은 '상품명'을 효과적으로 보정

```
%%time
train['형태소_상품명'] = train['상품명'].apply(lambda x: ' '.join(hannanum.nouns(x)))
```

```
Wall time: 1min 35s
```

```
%%time
test['형태소_상품명'] = test['상품명'].apply(lambda x: ' '.join(hannanum.nouns(x)))
```

```
Wall time: 8.6 s
```

Example

'테팔 청소기 전용 철제 거치대' → '테팔', '청소기', '전용', '철제', '거치대'

노출(분)	마더코드	상품코드	상품명	상품군	판매단가	취급액	성별	지불	...	네이버쇼핑_상품명	네이버쇼핑_리뷰갯수	네이버쇼핑_평점	네이버쇼핑_가격	형태소_상품명	가장_유사한_이름의 인덱스	가장_유사한_이름	유사도	유사상품_취급액	유사상품_판매단가
.000000	100030	200045	테팔 청소기 전용 철제 거치대	생활용품	69000	NaN	3	3	...	테팔 청소기 전용 철제 거치대	NaN	NaN	69000.0	테팔 청소기 전용 철제 거치대	30847	테팔 에어 포스 라이트 무선 청소기	[[0.3274713174606393]]	22438000	119000

3 데이터 전처

```
# train > 기상데이터 불러오기
weather_train = pd.read_csv('../data/weather_train_raw.csv',encoding='euc-kr')
weather_train.head()
```

```
# test > 기상데이터 불러오기
weather_test = pd.read_csv('../data/weather_test_raw.csv',encoding='euc-kr')
weather_test.head()
```

```
# train > 미세먼지데이터 불러오기
dust_train = pd.read_csv('../data/dust_train_raw.csv',encoding='euc-kr')
dust_train.head()
```

```
# test > 미세먼지데이터 불러오기
dust_test = pd.read_csv('../data/dust_test_raw.csv',encoding='euc-kr')
dust_test.head()
```

기상청 기상자료개방포털 보다나은정부

Train Data, Test Data에 '기온, 강수량, 풍속, 습도, 적설' Column 추가

기상청 홈페이지에 있는 기상자료개방포털에서 해당 데이터를 직접 다운로드

각 지역별로 인구비중에 따른 가중치를 두어 전국적인 기상정보에 관하여 판단

※ 단, 지역은 전국 시,도에서 인구가 가장 많은 지역을 하나씩 선택

※ 예외: 세종특별자치시는 데이터의 측정이 모두 이뤄지지 않아 불균형하고 대전과 가까워 제외

서울특별시	부산광역시	대구광역시	인천광역시
서울	부산	대구	인천
광주광역시	**대전광역시**	**울산광역시**	**경기도**
광주	대전	울산	수원
강원도	**충청북도**	**충청남도**	**전라북도**
원주	청주	천안	전주
전라남도	**경상북도**	**경상남도**	**제주특별시**
여수	포항	창원	제주

3 데이터 전처

Train Data, Test Data에 '기상' Column 전처리

Train, Test Data에서 기온 데이터의 결측값은 이전 시간대의 기온을 그대로 사용

∵ 기온은 이전 시간대의 영향을 받음.

Train, Test Data에서 풍속, 습도 데이터의 결측값은 평균값으로 사용

∵ 풍속, 습도는 하루 평균 수치에 관한 영향을 많이 받음.

Train, Test Data에서 강수량, 적설 데이터의 결측치는 Binary로 처리

∵ 강수량, 적설 데이터의 결측치 비율이 높음.

```
# train > 기온 결측값 -> ffil로 대체
weather_train[weather_train['기온(° C)'].isnull()]
```

```
# train > 풍속, 습도 결측값 -> 평균값으로 대체
wind_means_train = dict()
humid_means_train = dict()
for region in list(weather_train['지점명'].unique()):
    wind_means_train[region] = weather_train[weather_train['지점명']==region]['풍속(m/s)'].mean()
    humid_means_train[region] = weather_train[weather_train['지점명']==region]['습도(%)'].mean()
```

```
# train > 강수량, 적설 결측값 -> binary로 바꿔주기
weather_train['강수량(mm)'] = weather_train['강수량(mm)'].fillna(0)
weather_train['적설(cm)'] = weather_train['적설(cm)'].fillna(0)
weather_train['비'] = weather_train['강수량(mm)'].map(lambda x: 1 if x != 0 else 0)
weather_train['눈'] = weather_train['적설(cm)'].map(lambda x: 1 if x != 0 else 0)
```

```
dust_train.isnull().sum()
```

```
지점                          0
지점명                         0
일시                          0
1시간평균 미세먼지농도(㎍/㎥)        0
dtype: int64
```

```
dust_test.isnull().sum()
```

```
지점                          0
지점명                         0
일시                          0
1시간평균 미세먼지농도(㎍/㎥)        0
dtype: int64
```

Train Data, Test Data에 '미세먼지' Column 전처리

Train, Test Data에서 미세먼지 데이터의 결측값 존재 X

∴ 그대로 이용

3 데이터 전처

```
# train > 기상, 미세먼지 데이터 합쳐주기
weather_final_train = pd.merge(weather_train, dust_train, how = 'left', on = ['일시', '지점명'])
weather_final_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140050 entries, 0 to 140049
Data columns (total 9 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   지점명                 140050 non-null  object
 1   일시                  140050 non-null  datetime64[ns]
 2   기온(° C)              140050 non-null  float64
 3   풍속(m/s)             140050 non-null  float64
 4   습도(%)               140050 non-null  float64
 5   비                   140050 non-null  int64
 6   눈                   140050 non-null  int64
 7   지점                   58050 non-null   float64
 8   1시간평균 미세먼지농도(㎍/㎥)  58050 non-null   float64
dtypes: datetime64[ns](1), float64(5), int64(2), object(1)
memory usage: 10.7+ MB
```

```
# test > 기상, 미세먼지 데이터 합쳐주기
weather_final_test = pd.merge(weather_test, dust_test, how = 'left', on = ['일시', '지점명'])
weather_final_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11456 entries, 0 to 11455
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   지점명                 11456 non-null  object
 1   일시                  11456 non-null  datetime64[ns]
 2   기온(° C)              11456 non-null  float64
 3   풍속(m/s)             11456 non-null  float64
 4   습도(%)               11456 non-null  float64
 5   비                   11456 non-null  int64
 6   눈                   11456 non-null  int64
 7   지점                   4601 non-null   float64
 8   1시간평균 미세먼지농도(㎍/㎥)  4601 non-null   float64
dtypes: datetime64[ns](1), float64(5), int64(2), object(1)
memory usage: 895.0+ KB
```

Train Data, Test Data에 '기상' Columns + '미세먼지' Column

일시, 지점명이 같은 Value끼리 합침

But, 미세먼지 'Column'에서 결측치 발생

∵ '기상' Columns와 '미세먼지' Column의 관측 일시 차이

→ 결측치를 0으로 처리

```
# train > 미세먼지 농도 결측치 -> 0으로 채우기
weather_final_train['1시간평균 미세먼지농도(㎍/㎥)'] = weather_final_train['1시간평균 미세먼지농도(㎍/㎥)'].fillna(0)
```

```
# test > 미세먼지 농도 결측치 -> 0으로 채우기
weather_final_test['1시간평균 미세먼지농도(㎍/㎥)'] = weather_final_test['1시간평균 미세먼지농도(㎍/㎥)'].fillna(0)
```

```
weather_final_train.isnull().sum()
```

```
지점명                    0
일시                    0
기온(° C)                0
풍속(m/s)               0
습도(%)                 0
비                      0
눈                      0
1시간평균 미세먼지농도(㎍/㎥)    0
dtype: int64
```

```
weather_final_test.isnull().sum()
```

```
지점명                    0
일시                    0
기온(° C)                0
풍속(m/s)               0
습도(%)                 0
비                      0
눈                      0
1시간평균 미세먼지농도(㎍/㎥)    0
dtype: int64
```

3 데이터 전처

행정구역별 인구 데이터 불러오기

데이터 출처: 국가통계포털 행정구역별, 성별 인구수 데이터

인구 데이터 날짜: Train Data의 기준인 2019년

인구데이터를 통한 가중치 적용을 위한 인구비율 계산

```
# 행정구역별 인구데이터 받아오기
pop = pd.read_csv('../data/population2019_raw.csv',encoding='euc-kr')
pop.rename(columns={'행정구역':'지점명'},inplace=True)
pop
```

```
pop['지점명'] = pop['지점명'].map(lambda x : x.split('(')[0].strip())
pop.head()
```

	지점명	2019년_총인구수	2019년_세대수	2019년_세대당 인구	2019년_남자 인구수	2019년_여자 인구수	2019년_남여 비율
0	전국	51,849,861	22,481,466	2.31	25,864,816	25,985,045	1.00
1	서울특별시	9,729,107	4,327,605	2.25	4,744,059	4,985,048	0.95
2	부산광역시	3,413,841	1,497,908	2.28	1,675,417	1,738,424	0.96
3	대구광역시	2,438,031	1,031,251	2.36	1,205,286	1,232,745	0.98
4	인천광역시	2,957,026	1,238,641	2.39	1,482,249	1,474,777	1.01

```
# train > 시간대별로 더하기
weather_final_train = weather_train.groupby('일시').sum()
del weather_final_train['인구비율']
weather_final_train
```

```
pop['2019년_총인구수'] = pop['2019년_총인구수'].str.replace(',','').astype(int)
pop['인구비율'] = pop['2019년_총인구수']/(pop['2019년_총인구수'].sum())
```

'기상', '미세먼지' Column에 인구비율 가중치 적용

인구비율 값을 '기온', '풍속', '습도', '1시간평균 미세먼지 농도'에 곱함.

※ 단, '비', '눈'은 Binary로 처리하였으므로 인구비율 가중치 적용 X

```
# train > 각 지역별로 가중치(인구비율)를 곱해 계산
weather_train['기온(°C)'] = weather_train['기온(°C)']*weather_train['인구비율']
weather_train['비'] = weather_train['비']*weather_train['인구비율']
weather_train['풍속(m/s)'] = weather_train['풍속(m/s)']*weather_train['인구비율']
weather_train['습도(%)'] = weather_train['습도(%)']*weather_train['인구비율']
weather_train['눈'] = weather_train['눈']*weather_train['인구비율']
weather_train['1시간평균 미세먼지농도(㎍/㎥)'] = weather_train['1시간평균 미세먼지농도(㎍/㎥)']*weather_train['인구비율']
weather_train
```

3 데이터 전처

```
# 팔린 제품 수 계산하기(취급액 % 판매단가) -> 소수점 나올 수 있음
train['팔린제품수'] = train['취급액'] / train['판매단가']
# 소수점 첫째자리에서 반올림
train['팔린제품수'] = train['팔린제품수'].round()
train['팔린제품수']
```

```
0          53.0
1         110.0
2          82.0
3         174.0
4         167.0
          ...
37367      69.0
37368     286.0
37369      87.0
37370     621.0
37371     315.0
Name: 팔린제품수, Length: 37372, dtype: float64
```

Train Data에 '팔린제품수' Column 추가

'팔린제품수' Column = $\frac{\text{취급액 } Column}{\text{판매단가 } Column}$

※ 단, Test Data는 취급액 'Column이 없으므로 '팔린제품수' Column을 추가할 수 없음.

Train Data, Test Data에 '기업분류' Column 추가

'상품명' Column의 형태소로 구분하여 각 상품을 제조한 기업 단위 분류

중소기업	중견기업	대기업
Small	Midsize	Major

```
furniture['기업분류'] = pd.np.where(furniture.상품명.str.contains("유캐슬"), "Small",
                    pd.np.where(furniture.상품명.str.contains("이누스바스"), "Midsize",
                    pd.np.where(furniture.상품명.str.contains("보루네오"), "Small",
                    pd.np.where(furniture.상품명.str.contains("벨라홈"), "Small",
                    pd.np.where(furniture.상품명.str.contains("장수"), "Midsize",
                    pd.np.where(furniture.상품명.str.contains("장수흙침대"), "Midsize",
                    pd.np.where(furniture.상품명.str.contains("이조농방"), "Small",
                    pd.np.where(furniture.상품명.str.contains("삼익가구"), "Small",
                    pd.np.where(furniture.상품명.str.contains("한샘"), "Major",
                    pd.np.where(furniture.상품명.str.contains("레스토닉"), "Small", np.nan)))))))))

furniture.to_excel('../data/furniture.xlsx', encoding='cp949', index=False)
```

```
check_null(furniture)
```

3 데이터 전처

```
train['월'] = train['방송일시'].map(lambda x: x.month)
train['일'] = train['방송일시'].map(lambda x: x.day)
train['시'] = train['방송일시'].map(lambda x: x.hour)

test['월'] = test['방송일시'].map(lambda x: x.month)
test['일'] = test['방송일시'].map(lambda x: x.day)
test['시'] = test['방송일시'].map(lambda x: x.hour)
```

Train Data, Test Data에 '월', '일', '시' Column 추가

편성표를 짤 때 빠르게 조건을 구성하기 위하여 추가

Train Data, Test Data에 '공휴일' Column 추가

출근하지 않는 날의 T커머스 매출이 높다고 가정

공공데이터포털 API를 이용해 '공휴일', '토요일', '일요일'을 '공휴일'로 구분

평일	공휴일	토요일	일요일
0	1	1	1

```
# 요청 URL과 오퍼레이션
URL = 'http://apis.data.go.kr/B090041/openapi/service/SpcdeInfoService'
OPERATION = 'getRestDeInfo' # 국경일 + 공휴일 정보 조회 오퍼레이션
SERVICEKEY = 'G8L%2BFnmoZnMZY3Dx5cRcoAFCZHggIkT%2Ft%Ak2INeERrwHBxKkLtaRu1EbPu1SKzq%2BCz5X1G%2FI2eTT5LuKmIMug%3D%3D'

# 공휴일 리스트
holiday_list = []

for year in ['2019','2020']:
    for month in ['01','02','03','04','05','06','07','08','09','10','11','12']:
        PARAMS = {'solYear':year, 'solMonth':month}
        request_query = get_request_query(URL, OPERATION, PARAMS, SERVICEKEY)
        response = requests.get(url=request_query)
        root_element = ElementTree.fromstring(response.text)
        iter_element = root_element.iter(tag='item')
        for element in iter_element:
            holiday_list.append(element.find('locdate').text)

is_holiday = []
for idx in range(len(train)):
    if train.loc[idx,'요일(숫자)'] == 5 or train.loc[idx,'요일(숫자)'] == 6:
        is_holiday.append(1)
    elif str(train.loc[idx,'방송일시'])[:10].replace('-','') in holiday_list:
        is_holiday.append(1)
    else:
        is_holiday.append(0)

train['공휴일'] = is_holiday
```

3 데이터 전처

```
# input column 정의
drop_cols = ['방송일시','시청률_합','시청률_평균','요일','형태소_상품명', '가장_유사한_이름의인덱스',
             '가장_유사한_이름', '유사도', '유사상품_취급액', '유사상품_판매단가','네이버쇼핑_상품명','네이버쇼핑_리뷰갯수'
            ,'네이버쇼핑_평점','네이버쇼핑_가격','팔린제품수']
selected_cols = [col for col in list(train.columns) if col not in drop_cols]
label_cols = ['상품군','기업분류']
onehot_cols = ['월','일','시','성별','지물','상품군','요일(숫자)','비','눈','기업분류','공휴일']

Ttrain = train[selected_cols]
Ttest = test[selected_cols]
```

```
# 원핫인코딩
Ttrain = pd.get_dummies(Ttrain, columns=onehot_cols)
Ttest = pd.get_dummies(Ttest, columns=onehot_cols)

# 칼럼 일치
Ttest = pd.DataFrame(Ttest, columns=Ttrain.columns)
Ttest.fillna(0, inplace=True)
```

```
# 레이블 인코딩
for col in label_cols:
    le = LabelEncoder()
    le = le.fit(Ttrain[col])
    Ttrain[col] = le.transform(Ttrain[col])
    Ttest[col] = le.transform(Ttest[col])
```

'Label Encoding & One-hot Encoding

Label Encoding: feature value가 한글인 Column을 숫자로 바꿈

One-hot Encoding: Categorical 변수를 숫자 데이터로 바꿈

'상품명' Column Vector화

상품명이 취급액에 영향을 준다고 가정

```
# fit_on_texts: 각 단어에 고유의 정수를 부여함
tokenizer = Tokenizer()
tokenizer.fit_on_texts(word_train)
```

Train Data으로 학습시킨 Tokenizer를 Test Data에 적용

```
threshold = 10
total_cnt = len(tokenizer.word_index) # 단어의 수
rare_cnt = 0 # 등장 빈도수가 threshold보다 작은 단어의 개수를 카운트
total_freq = 0 # 훈련 데이터의 전체 단어 빈도수 총 합
rare_freq = 0 # 등장 빈도수가 threshold보다 작은 단어의 등장 빈도수의 총 합

# 단어와 빈도수의 쌍(pair)을 key와 value로 받는다.
for key, value in tokenizer.word_counts.items():
    total_freq += value

    # 단어의 등장 빈도수가 threshold보다 작으면
    if(value < threshold):
        rare_cnt += 1
        rare_freq += value

print('단어 집합(vocabulary)의 크기 :',total_cnt)
print('등장 빈도가 %s번 이하인 희귀 단어의 수: %s'%(threshold - 1, rare_cnt))
print("단어 집합에서 희귀 단어의 비율:", (rare_cnt / total_cnt)*100)
print("전체 등장 빈도에서 희귀 단어 등장 빈도 비율:", (rare_freq / total_freq)*100)
```

```
단어 집합(vocabulary)의 크기 : 2385
등장 빈도가 9번 이하인 희귀 단어의 수: 806
단어 집합에서 희귀 단어의 비율: 33.79454926624738
전체 등장 빈도에서 희귀 단어 등장 빈도 비율: 2.409611980789272
```

```
# 전체 단어 개수 중 빈도수 2이하인 단어 개수는 제거.
# 0번 패딩 토큰과 1번 OOV 토큰을 고려하여 +2
vocab_size = total_cnt - rare_cnt + 2
print('단어 집합의 크기 :',vocab_size)
```

```
단어 집합의 크기 : 1581
```

3 데이터 전처

```
hit = pd.read_excel('../data/hit.xlsx')
```

```
for i in tqdm_notebook(range(len(train))):
    for good in hit:
        if train['상품명'][i] == good:
            train['히트상품'][i] = 1
```

Train Data, Test Data에 '히트상품' Column 추가

Data에 대하여 '상품군' Column별로 '취급액' Column의 Value가 **각 월별 상위 5위인 상품을 히트상품으로 취급**

※ 단, 상위 5위 내 상품의 상품명이 동일하더라도 방송일시가 다르면 다른 상품으로 취급

문장 Vector화를 진행시켰을 때 Test Data에서 코사인 유사도가 0.8 이상이면 '히트상품' Column 1

코사인 유사도	0.8 이하	0.8 초과
히트상품	0	1

```
sentence = pd.concat([train['형태소_상품명'],test['형태소_상품명']])
sentence.index = range(len(train) + len(test))
sentence
```

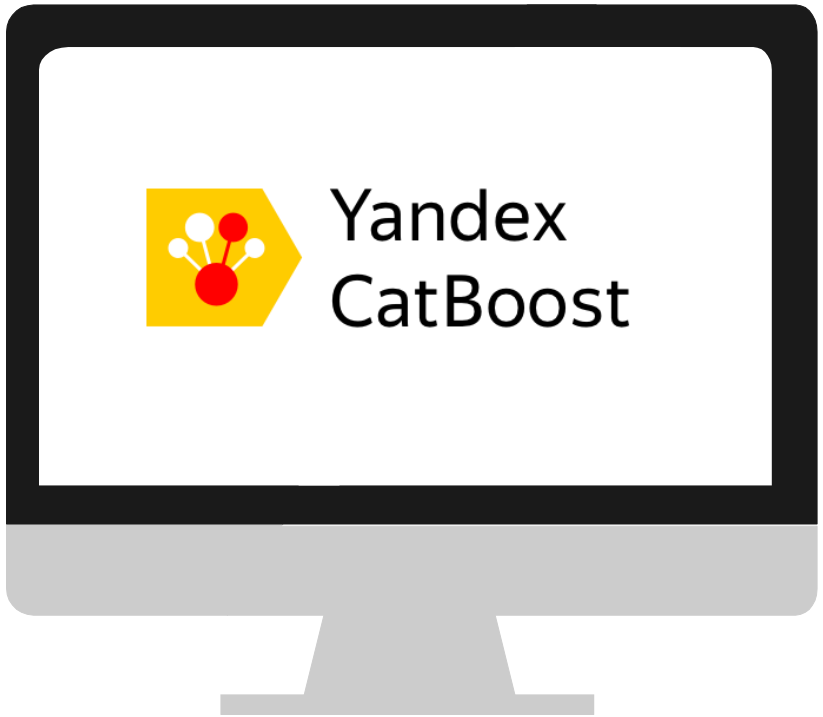
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# 객체 생성
tfidf_vectorizer = TfidfVectorizer()
# 문장 벡터화 진행
tfidf_matrix = tfidf_vectorizer.fit_transform(sentence)
# 각 단어
text = tfidf_vectorizer.get_feature_names()
# 각 단어의 벡터 값
idf = tfidf_vectorizer.idf_
```

```
noun_hit = list()
for h in hit:
    noun_hit.append(' '.join(hannanum.nouns(h)))
```

```
from konlpy.tag import Hannanum
hannanum = Hannanum()
```

```
adict = dict()
for noun in tqdm_notebook(noun_hit):
    for i in range(len(train)):
        if train['형태소_상품명'][i] == noun:
            adict[noun] = i
            break
```

4 데이터 분석



CatBoost

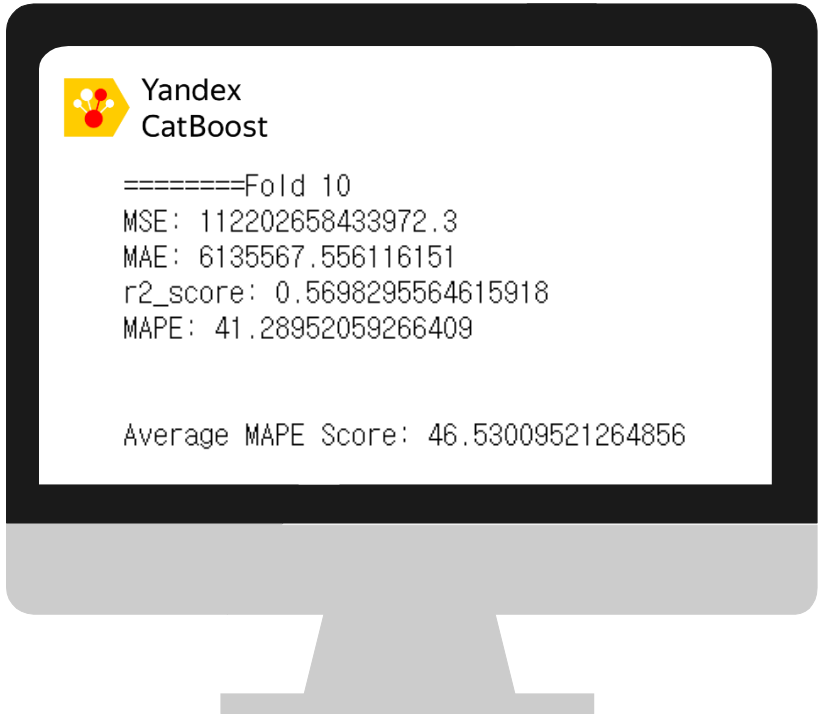
1. **기존의 부스팅 방법의 문제점 해결**
   느린 학습 속도 & Overfitting

2. **일부 Train Data로만 잔차계산 진행**
   잔차계산에 사용하지 않은 Data는 모델로 예측한 값을 사용

3. **Data를 Shuffling하여 선택**
   단, 데이터를 일부만 선택하여 Shuffling을 진행

Data를 일부만 Shuffling하여 선택하므로
Overfitting 문제 방지할 수 있어 사용

4 데이터 분석



CatBoost

```
cat_params = {
    'iterations':1000,
    'learning_rate':0.05,
    'depth':16,
    'loss_function':'MAE',
    'eval_metric':'MAPE',
    'thread_count':16,
    'rsm':0.95,
    'bagging_temperature':0.8
}

for i,(train_ind, test_ind) in enumerate(KF.split(Ttrain)):
    print('========Fold',i+1)
    Xtrain, XCV, ytrain, yCV = Ttrain[train_ind], Ttrain[test_ind], Ttarget[train_ind], Ttarget[test_ind]

    cat_model = cat.CatBoostRegressor(**cat_params)

    cat_model.fit(Xtrain,ytrain,eval_set=[(XCV,yCV)],early_stopping_rounds=30,verbose=False)

    pred = cat_model.predict(XCV)

    print_score(pred,yCV)

    cat_preds += cat_model.predict(Ttest)/folds
    cat_avg += mape(pred,yCV)/folds

print('₩n₩nAverage MAPE Score:',cat_avg)
```

	iterations	depth	rsm	MAPE
결과	1000	16	0.95	**46.53**

4 데이터 분석

XGBoost

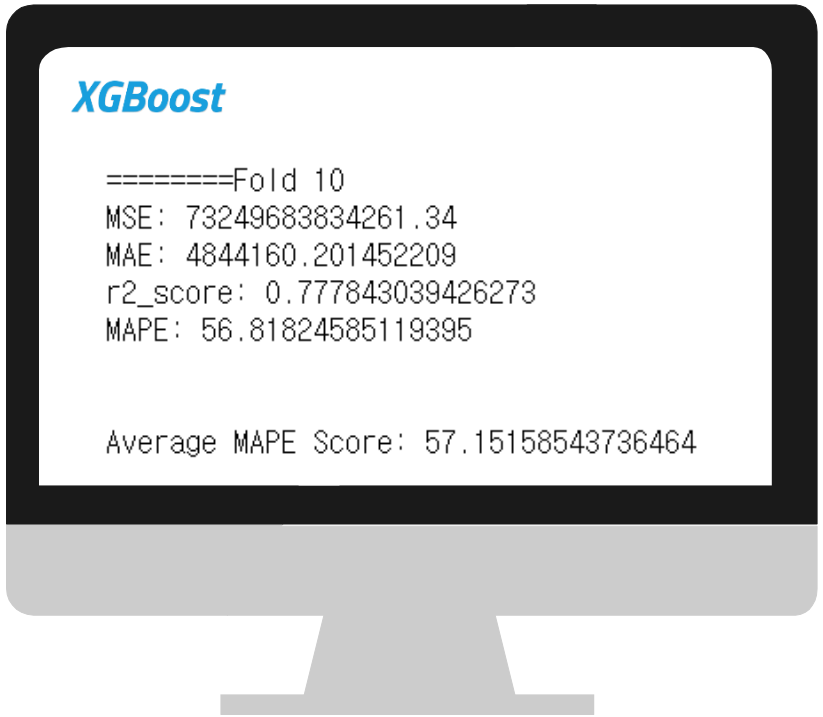
1 **Parallel Processing**
GBM의 단점인 느린 속도를 병렬처리를 통해 극복

2 **Custom Optimization**
목적과 평가 기준을 자유롭게 할 수 있음

3 **Tree Pruning**
max_depth까지만 Split하고 pruning 진행
Positive gain이 없는 node를 제거

Custom Optimization을 할 수 있다는 점에서 사용!

4 데이터 분석



XGBoost

```
xgb_params = {
    'n_estimators':1000,
    'learning_rate':0.05,
    'gamma':0,
    'subsample':0.75,
    'max_depth':10,
}

def xgb_mape(preds, dtrain):
    labels = dtrain.get_label()
    return np.mean(np.abs((labels - preds) / (labels)))*100

for i,(train_ind, test_ind) in enumerate(KF.split(Ttrain)):
    print('========Fold',i+1)
    Xtrain, XCV, ytrain, yCV = Ttrain[train_ind], Ttrain[test_ind], Ttarget[train_ind], Ttarget[test_ind]

    xgb_model = xgb.XGBRegressor(**xgb_params)

    xgb_model.fit(Xtrain,ytrain,eval_set=[(XCV,yCV)],eval_metric=['rmse','mae'],early_stopping_rounds=30,verbose=False)

    pred = xgb_model.predict(XCV)

    print_score(pred,yCV)

    xgb_preds += xgb_model.predict(Ttest)/folds
    xgb_avg += mape(pred,yCV)/folds


print('₩n₩nAverage MAPE Score:',xgb_avg)
```

	n_estimators	max depth	sub sample	MAPE
결과	1000	10	0.75	**57.15**

4 데이터 분석

LightGBM

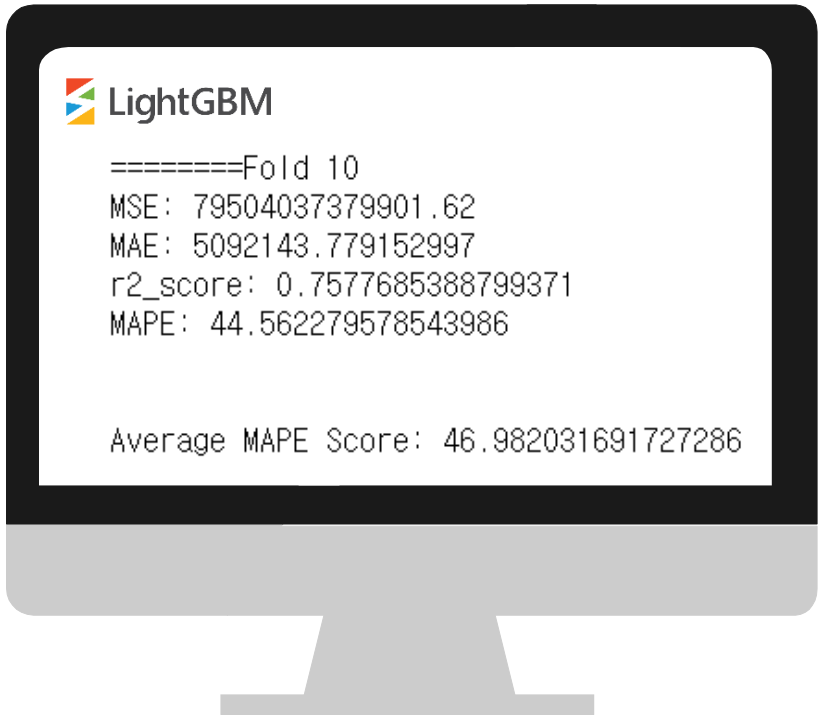
1. **Leaf-wise 트리 분할 사용**
   비대칭적이고 깊은 트리가 생성되지만 level-wise보다 손실을 줄일 수 있음.

2. **Categorical Feature의 자동 변환**
   One-hot-Encoding을 사용하지 않아도 최적 분할 가능

3. **메모리를 적게 차지하고 속도가 빠름**
   XGBoost에 비하여 속도가 빨라서 최적화된 모델 구현 가능

Train Data가 37000여 개이므로 유일한 단점인 Overfitting 문제가 해결되어 사용!

4 데이터 분석



LightGBM

```
lgb_params = {
    'n_estimators':1000,
    'num_leaves':512,
    'learning_rate':0.05,
    'max_depth':100,
    'boosting_type':'gbdt',
    'n_jobs':-1
}

for i,(train_ind, test_ind) in enumerate(KF.split(Ttrain)):
    print('========Fold',i+1)
    Xtrain, XCV, ytrain, yCV = Ttrain[train_ind], Ttrain[test_ind], Ttarget[train_ind], Ttarget[test_ind]

    lgb_model = lgb.LGBMRegressor(**lgb_params)

    lgb_model.fit(Xtrain,ytrain,eval_set=[(XCV,yCV)],eval_metric='mape',early_stopping_rounds=30,verbose=False)

    pred = lgb_model.predict(XCV)

    print_score(pred,yCV)

    lgb_preds += lgb_model.predict(Ttest)/folds
    lgb_avg += mape(pred,yCV)/folds

print('₩n₩nAverage MAPE Score:',lgb_avg)
```

	n_estimators	num_leaves	max depth	MAPE
결과	1000	512	100	**46.98**

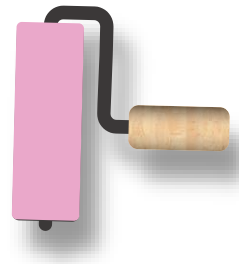
4 데이터 분석

Test Data 취급액 반영하여 판매 실적 예측 파일 생성

```
count = 0
for idx in range(len(submission)):
    if submission.loc[idx,'상품군'] == '무형':
        submission.loc[idx,'취급액'] = 0
        continue
    submission.loc[idx,'취급액'] = total_preds[count]
    count+=1
    if count==2716:
        break
```

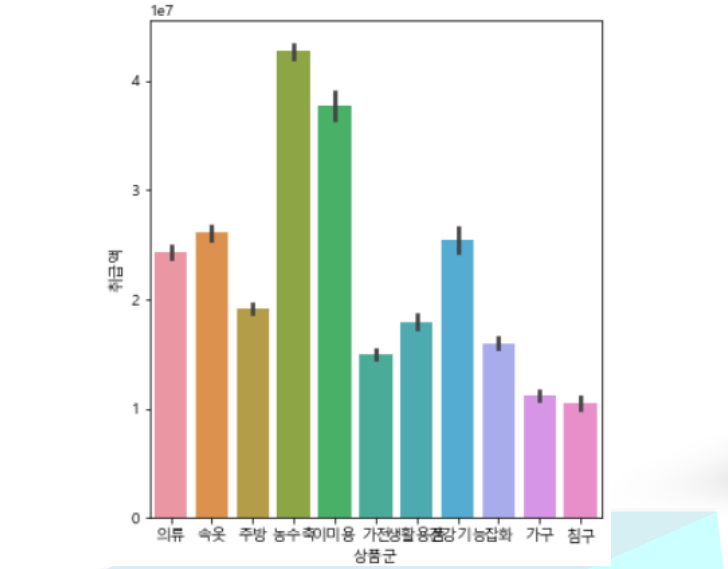
```
submission.to_excel('../data/final_submission.xlsx',index=False)
```

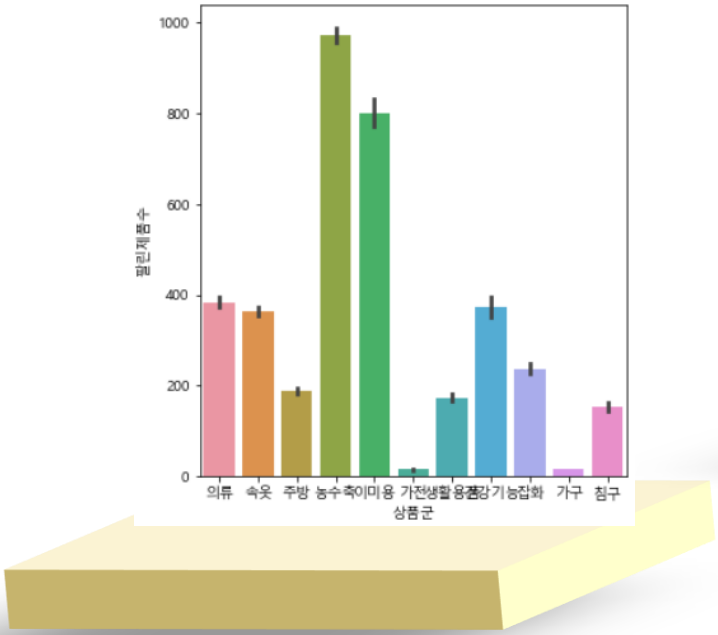
Excel 파일에 저장

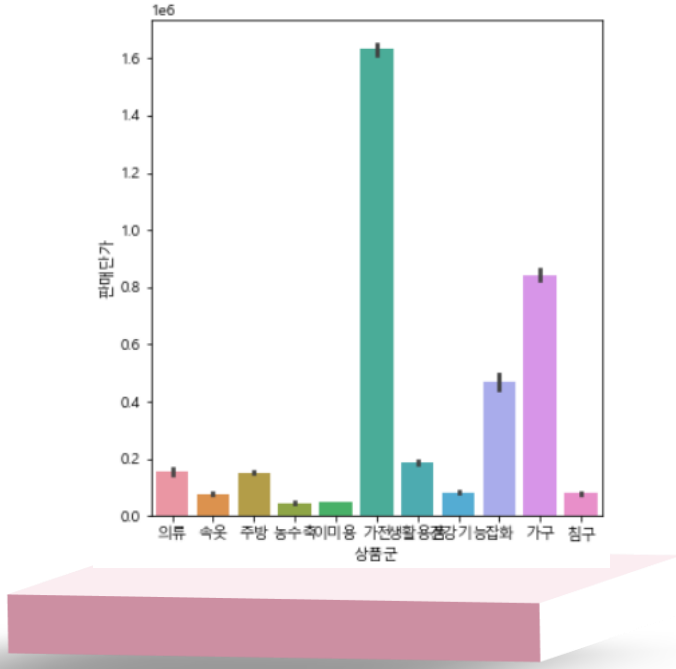


4 데이터 분석

판매단가가 낮고 취급액이 높고 팔린 제품 수가 많은 상품군:
농수축, 이미용, 건강기능







위 상품군에 대한 히트상품을 중심으로 최적 편성표를 짤 필요 有

4 데이터 분석

농수축 매출 TOP 제품들

농수축의 히트상품: 손질갑오징어,
손질왕꼬막, 소곱장전골

```
[('국내산 손질갑오징어 8팩', 12),
('우리바다 손질왕꼬막 24팩', 5),
('AAB의 소곱창전골 800g x 8팩', 4),
('유귀열의 The 귀한 능이 누룽지삼계탕 8팩', 4),
('피시원 국내산 절단 햇 꽃게 7팩', 4),
('피시원 국내산 절단 햇 꽃게 2.4kg', 4),
('국내산 손질 햇 갑오징어 9팩', 4),
('자연산 손질 통오징어 21미', 3),
('안동간고등어 20팩', 3),
('옛날 그 쥐포 110장 (11팩)', 2),
('피시원 국내산 햇 손질문어 8팩', 2),
('피시원 국내산 햇 손질문어 7팩', 2),
```

이미용 매출 TOP 제품들

이미용의 히트상품: 남성기초세트,
젤네일스트립 set

```
[('비버리힐스폴로클럽 남성기초세트(골드+플라)', 14),
('비버리힐스폴로클럽 남성기초세트(2018FW골드)', 10),
('비버리힐스폴로클럽 남성기초세트(골드+콜라겐)', 8),
('바바코코 젤네일스트립 SET', 6),
('[VONIN]보닌 남성기초세트', 6),
('참존 탑뉴스 지이링클 기초세트', 6),
('라라츄 틴트 아이브로우 바', 5),
('비버리힐스폴로클럽 남성기초세트(플라센타아이크림+선크림)', 4),
('스포메틱스 네츄럴 선스틱', 1)]
```

건강기능 매출 TOP 제품들

건강기능의 히트상품: 생유산균골드,
다이어트식품, 석류즙

```
[('일시불 종근당건강 락토핏 생유산균골드 단하루', 22),
('  뉴트리원 다이어트 구미', 10),
('정직한 농부의 석류', 9),
('통째로 착즙한 루비 석류즙 84포', 5),
('(직매입) 종근당건강 아이클리어 루테인', 4),
('닥터 슈퍼 루테인', 4),
('[광동제약] 석류 100, 총 5박스', 3),
('제주농장 유기 양배추진액 120포', 2),
('무이자 종근당건강 락토핏 생유산균골드 단하루', 1)]
```

4 데이터 분석

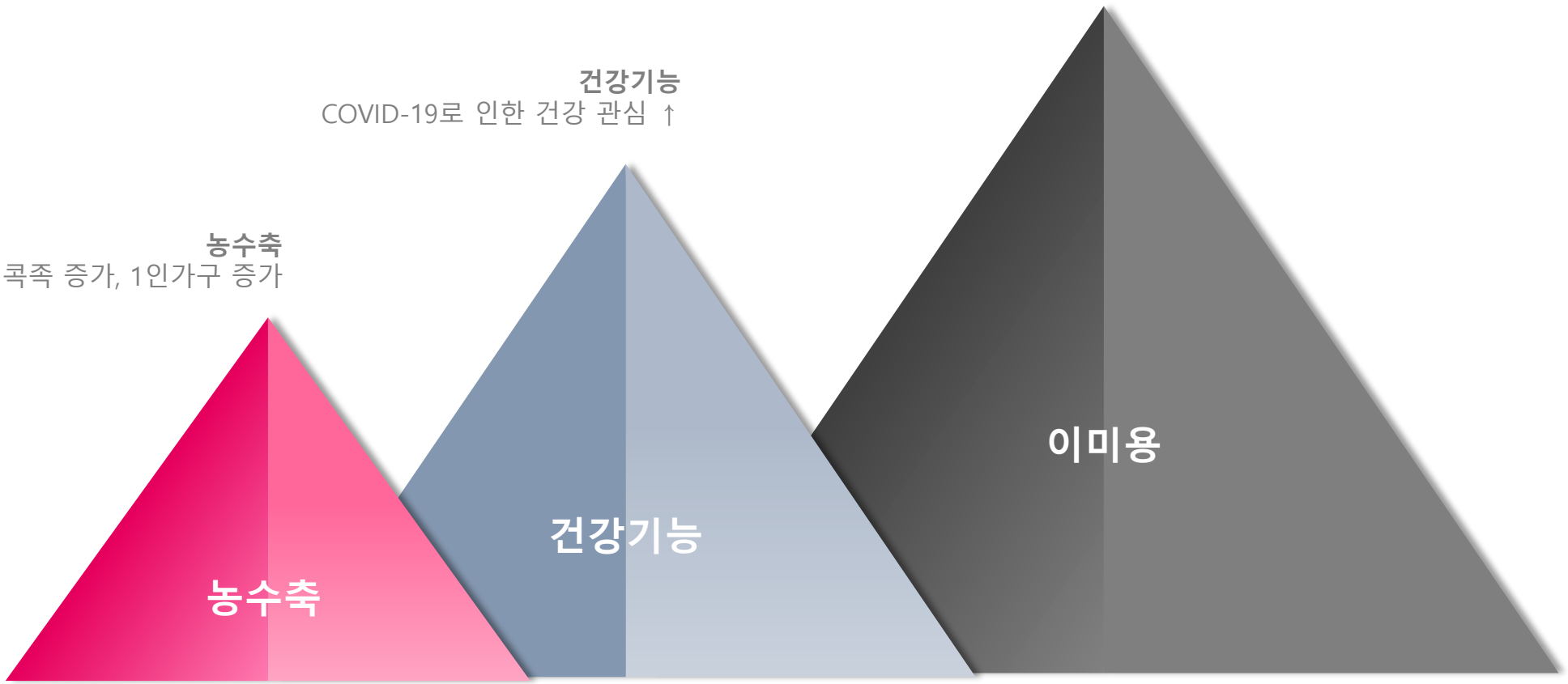
출처: PART 3. NS홈쇼핑_ 상반기 TOP10 상품에 식품 7개 랭크

김선희 기자 | 승인 2020.08.03 17:17 | 댓글 0

이미용
주된 상품: 남성기초세트
색조화장에 대한 판매전략보다는 피부 기초상품에 대한 판매전략 필요

건강기능
COVID-19로 인한 건강 관심 ↑

농수축
COVID-19로 인한 집콕족 증가, 1인가구 증가

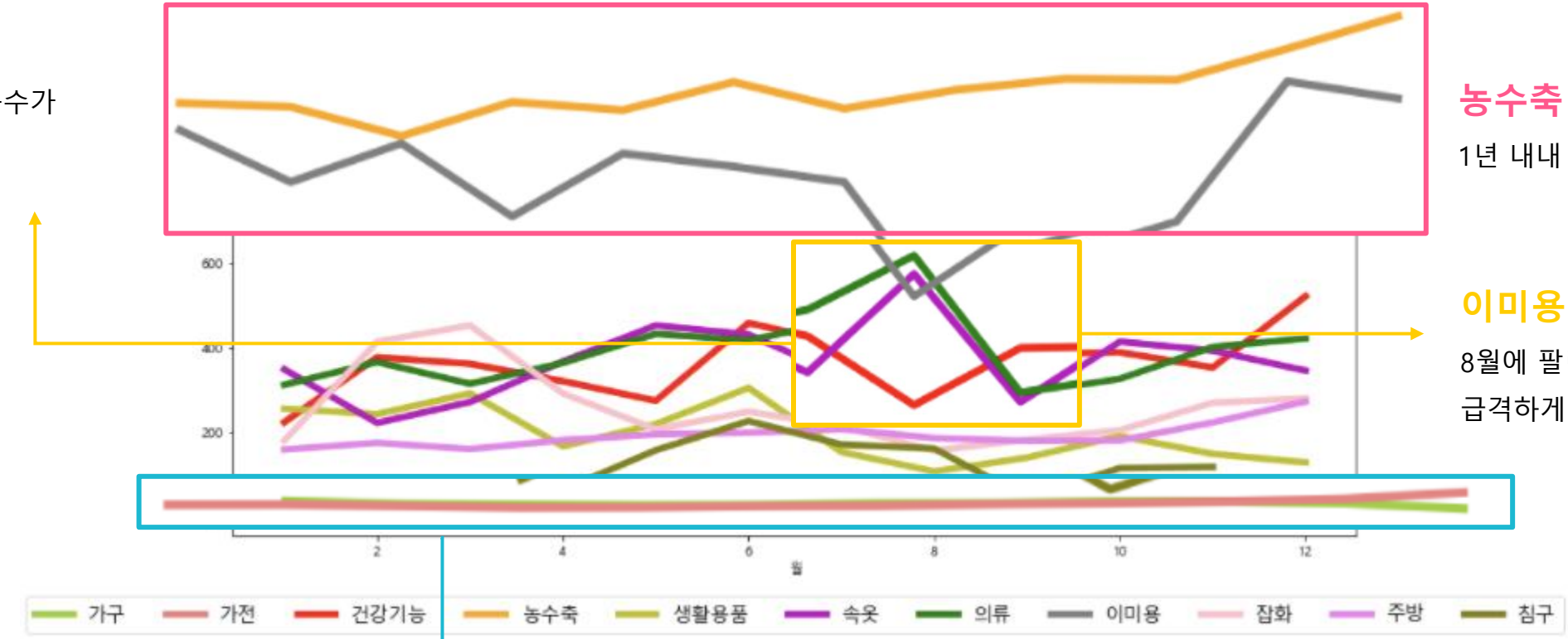


4 데이터 분석

의류, 속옷

8월에 팔린 제품수가

급격하게 증가



농수축

1년 내내 꾸준하게 상위를 유지

이미용

8월에 팔린 제품수가

급격하게 감소

월별 상품군별 팔린제품수를 나타낸 그래프 / X축: 월, Y축: 팔린제품수

가전, 가구

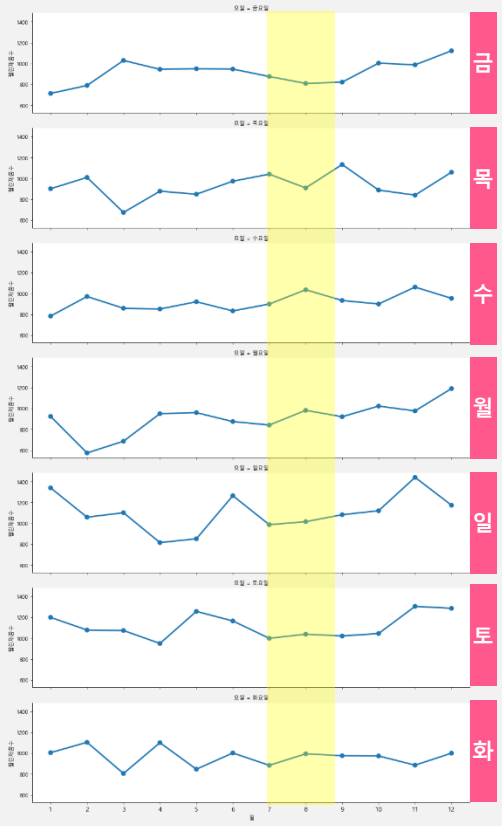
팔린 제품수가 1년 내내 하위권

8월 달에는 편성표를 짤 때 이미용 비율을 줄이고, 의류, 속옷 비율을 높이는 것을 추천

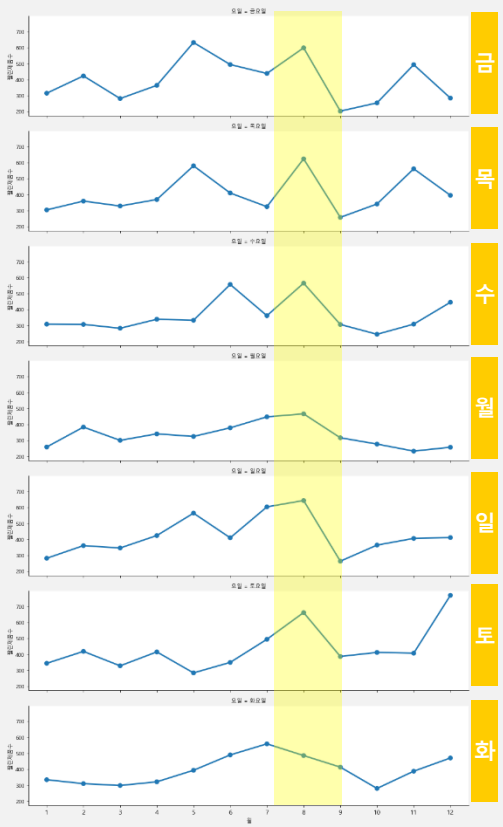
4 데이터 분석

요일별 월별 상품군별 팔린제품수를 나타낸 그래프 / X축: 월, Y축: 팔린제품수

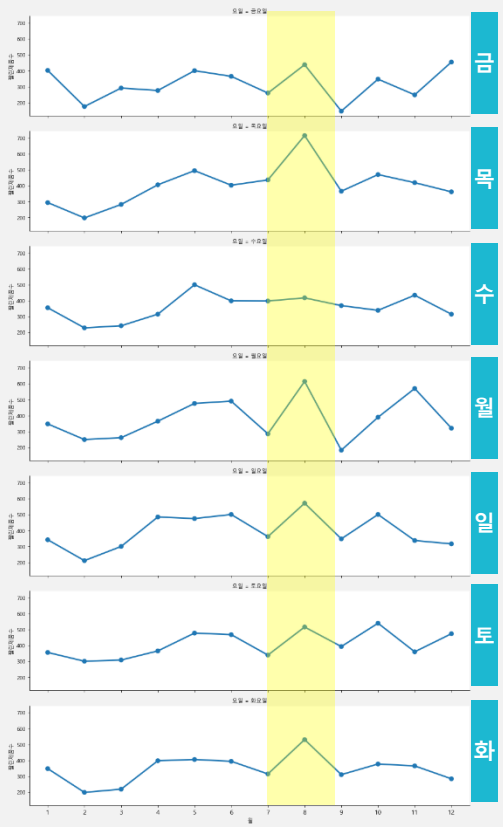
농수축



의류



속옷



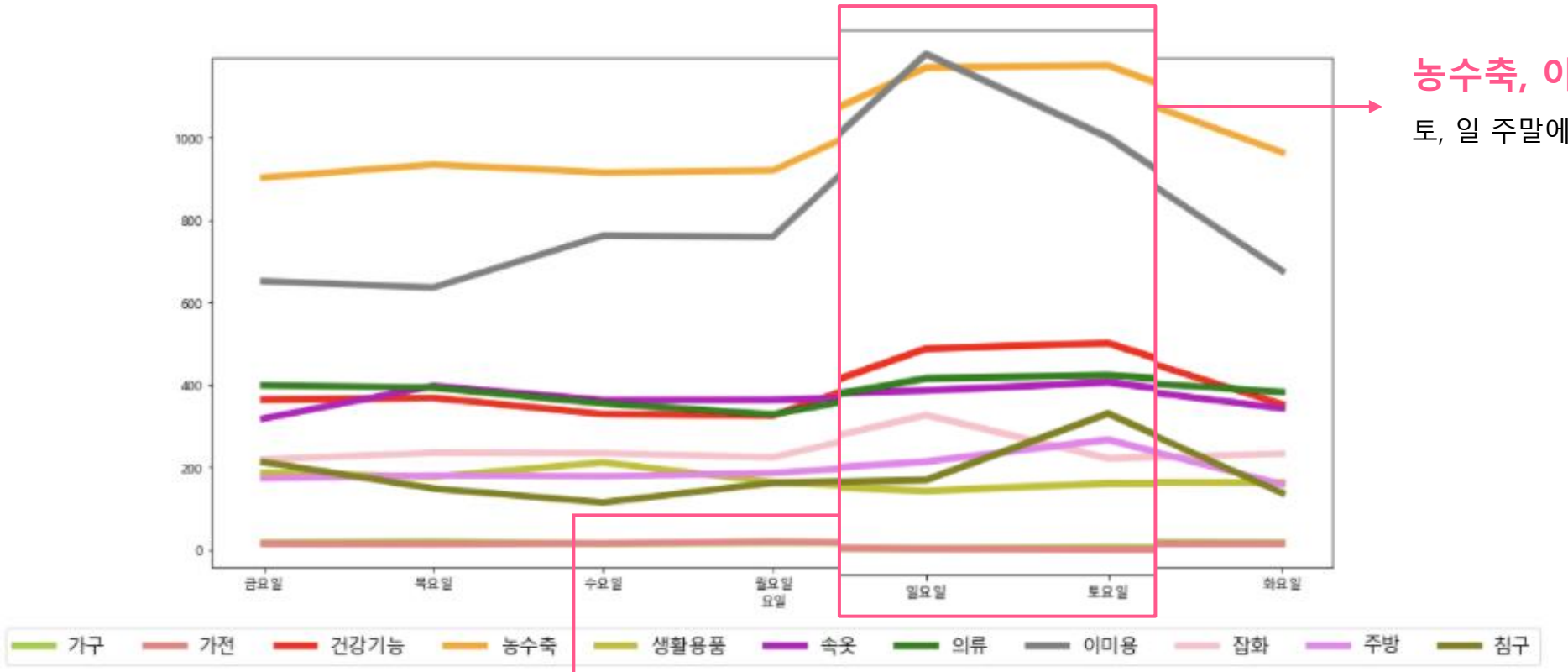
의류, 속옷

8월 매출
모든 요일 ↑

매출전략

상이한
매출전략
8월에
수립 필요!

4 데이터 분석



농수축, 이미용 상품군

토, 일 주말에 판매량 크게 ↑

요일별 상품군별 팔린제품수를 나타낸 그래프 / X축: 월, Y축: 팔린제품수

다른 상품군

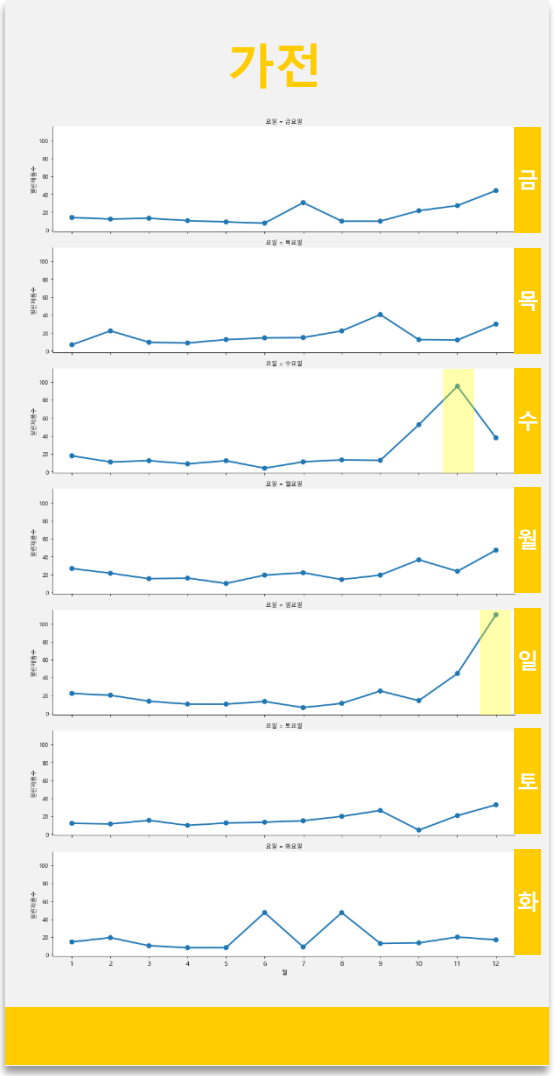
농수축, 이미용보단 적어도

토, 일 주말에 판매량 ↑

∴ **적절한 배치 필요**

대부분 상품군에서 주말 판매량 ↑

4 데이터 분석



급격한 판매량 보인 월별 요일
11월: 수요일, 12월: 일요일, 6월 및 8월: 화요일

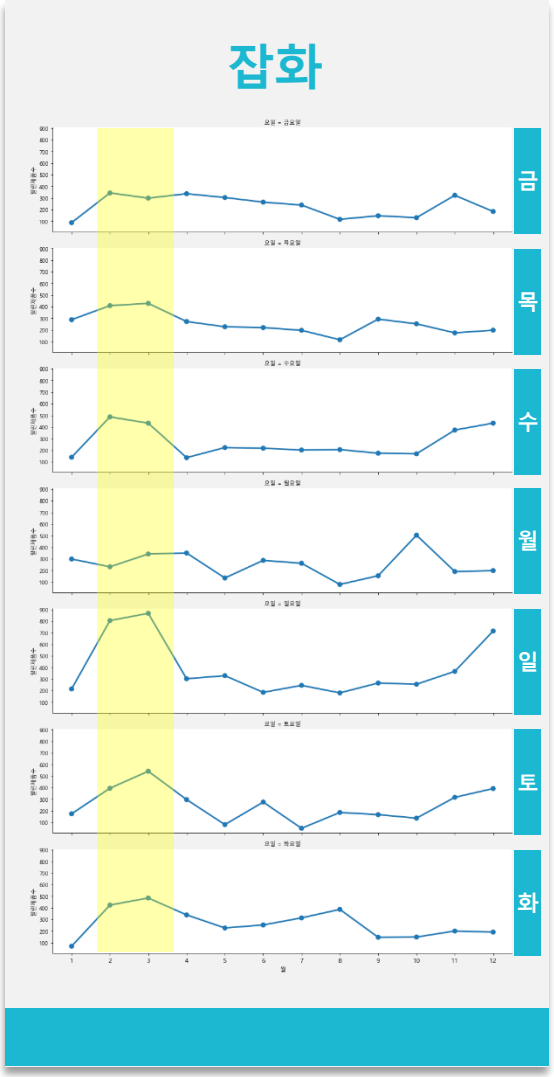
가전

여름철: 에어컨, 세탁기 판매량 ↑
겨울철: 냉장고 판매량 ↑

∴ 가전: 계절성을 가지고 있으므로
계절별 상품의 배치를 적절하게 할 필요 ↑

가전에 한하여 요일별 월별 상품군별 팔린제품수를 나타낸 그래프 / X축: 월, Y축: 팔린제품수

4 데이터 분석



잡화

2,3월 매출: 대부분의 요일에서 가장 높음.
특히, 일요일 매출 ↑

```
[('칼리베이직 여성백 3종세트', 6),
 ('도스둔도스 카이만 엠보 소가죽 핸드백 1종', 5),
 ('아가타 썸머 샌들세트', 5),
 ('아가타 꽃든 마스터 2종 (펌프스1종+플랫슈즈1종)', 5),
 ('2019 S/S 엘리자베스아덴 선글라스 2종', 4),
 ('썸익스 클레오 여행가방 9종세트', 4),
 ('[투이띠에] 18K 베르니아 주얼리 세트', 4),
 ('오엘로 여성 겨울모자 3종', 4),
 ('2019 S/S 로베르타 디 까메리노 선글라스', 2),
 ('AAD 소가죽 투웨이 샌들', 2),
 ('아가타 소가죽 데이쿠션로퍼', 2),
 ('칼리베이직 2019 패브릭 백 4종 (시즌 3)', 2),
 ('오엘로 겨울모자 3종 (2차)', 2),
 ('일시불[가이거] 제니스시계 주얼리세트', 1),
 ('스프리스 여성 코트 스니커즈', 1),
 ('2019 S/S 기라로쉬 올인원 선글라스', 1),
 ('일시불 썸익스 트로이카 여행가방 10종세트', 1),
 ('도스둔도스 카이만 엠보 소가죽 핸드백 1종(역시즌)', 1),
 ('생홍황 14K 글로리아 목걸이 컬렉션 8종', 1),
 ('레노마 우/양산 2종세트', 1),
 ('생홍황 18K 베니스 주얼리 컬렉션 11종', 1),
 ('시스마르스 플렉시블 웨지 펌프스', 1),
 ('아가타 윈터 에디션(부츠1종+소프트퍼슈즈1종)', 1),
 ('[RYN] 린 남성 다이얼락 히트 원 퍼 방한화 2종', 1),
 ('[RYN] 린 여성 다이얼락 히트 원 퍼 방한화 2종', 1),
 ('구찌 더블G 오버 선글라스', 1)]
```

여성용 핸드백의 상품이
잡화 주요 상품: 여성용 핸드백
여성 Target 상품: 매출의 거의 대부분 차지

특히, 코로나19 이후에도 명품 여성핸드백
이
매출 주력상품으로 자리잡음을 확인할 수
있다.

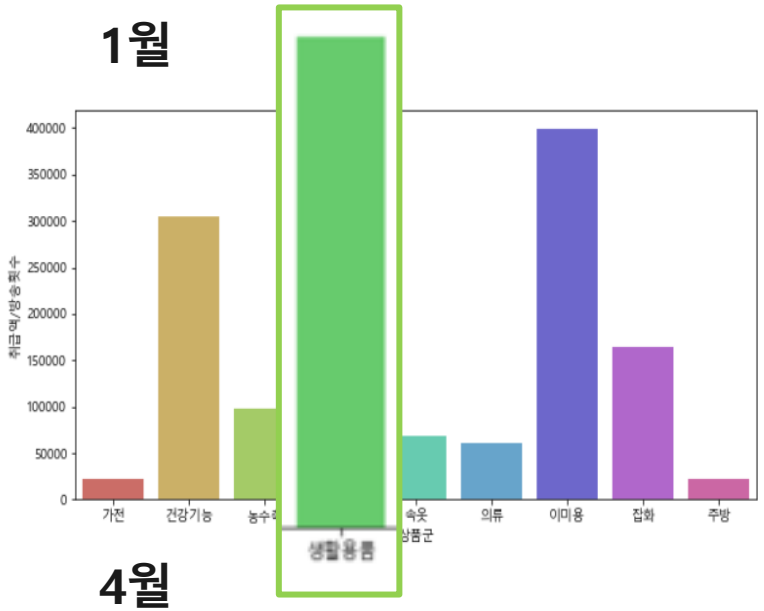
https://biz.chosun.com/site/data/html_dir/2020/06/19/2020061904046.html

코로나 사태에도 잘 팔리는 명품 핸드백…지난해 13% 가격 상승
biz.chosun.com

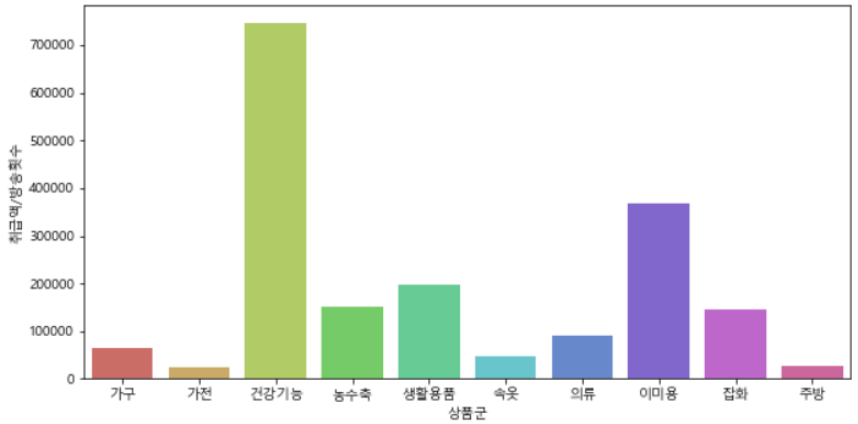
잡화에 한하여 요일별 월별 상품군별 팔린제품수를 나타낸 그래프 / X축: 월, Y축: 팔린제품수

4 데이터 분석

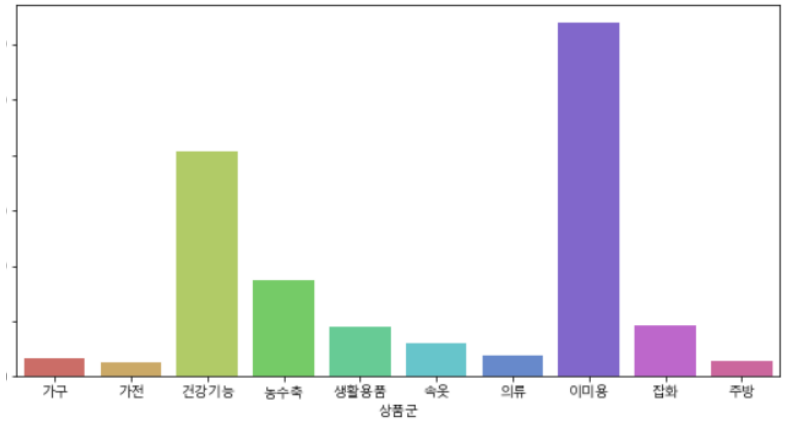
1월



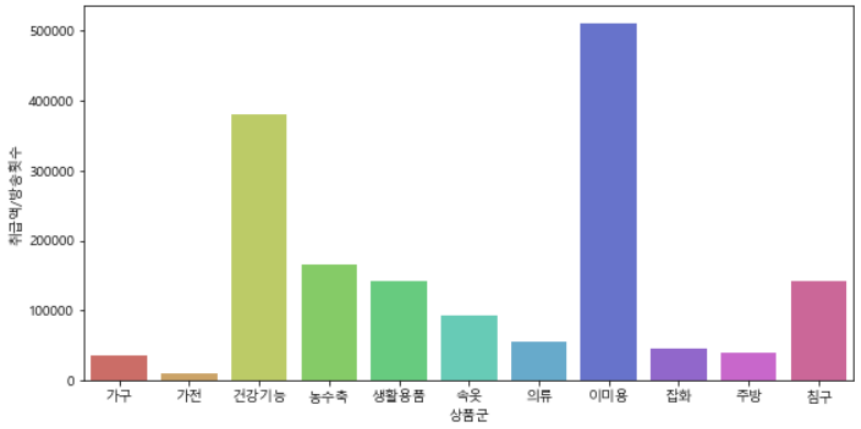
2월



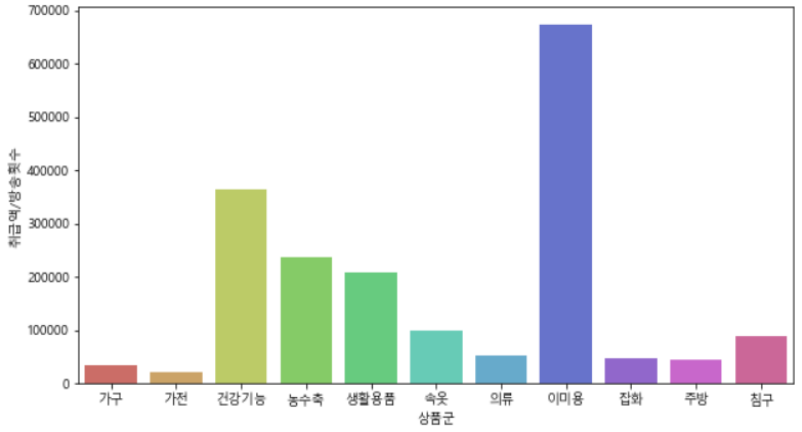
3월



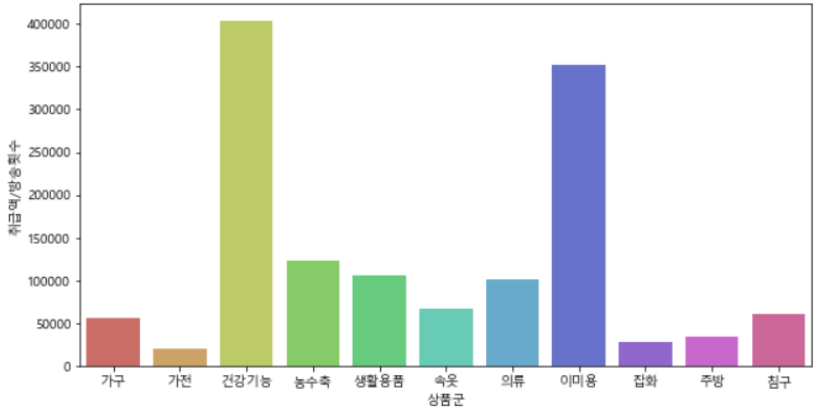
4월



5월

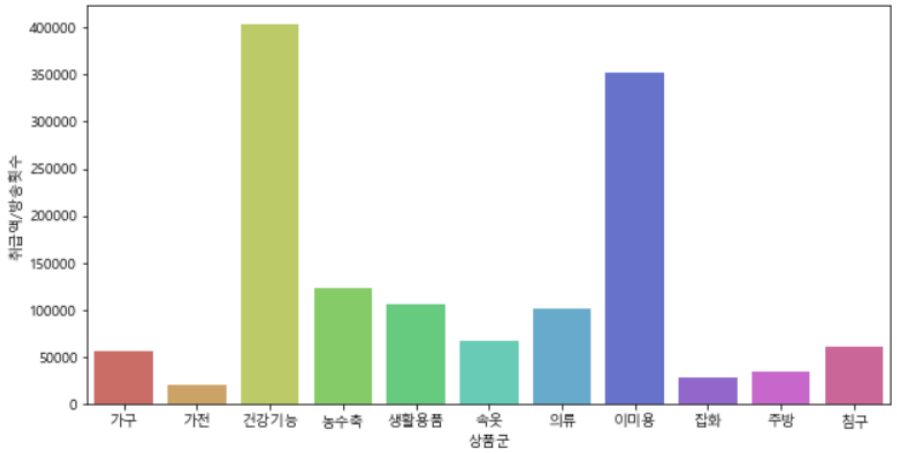


6월

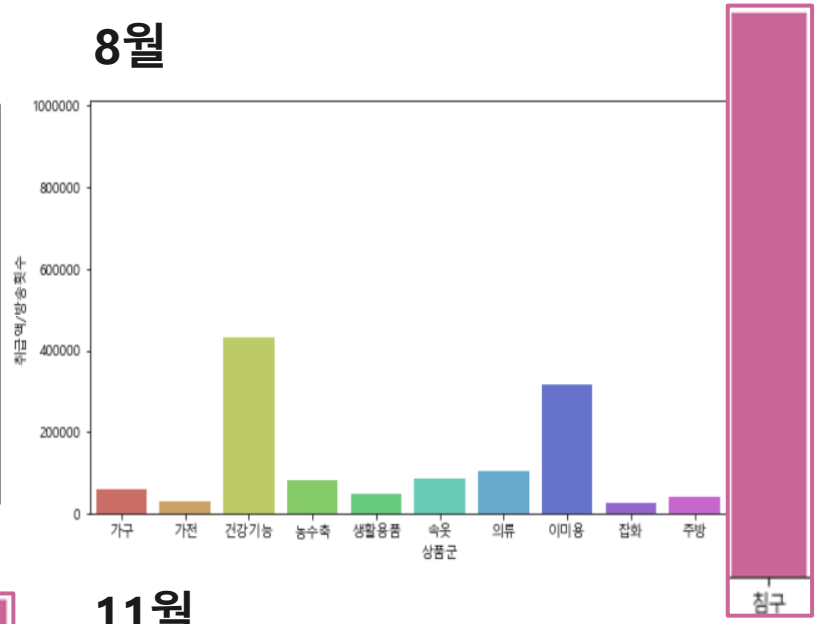


4 데이터 분석

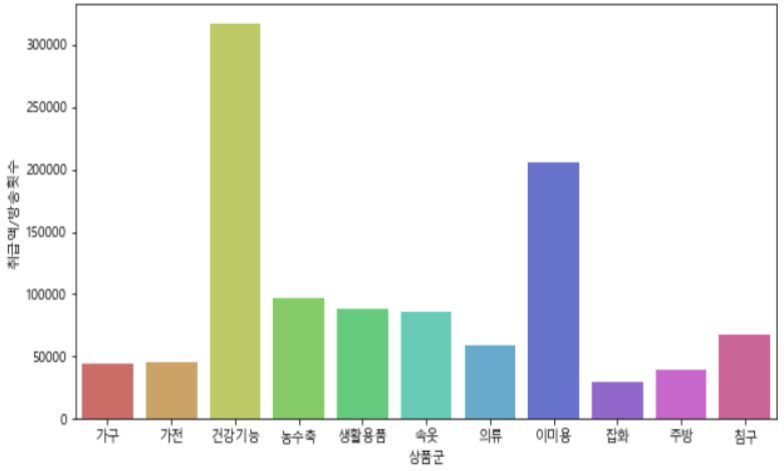
7월



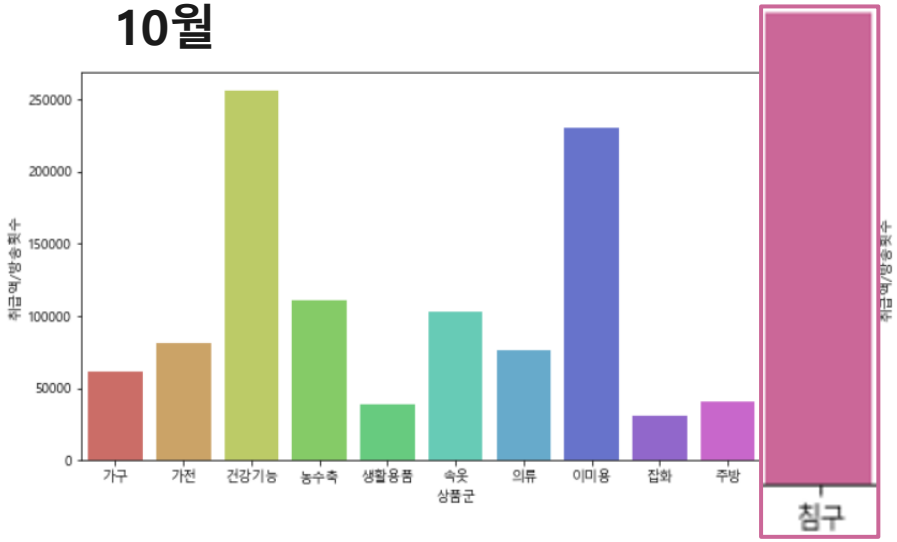
8월



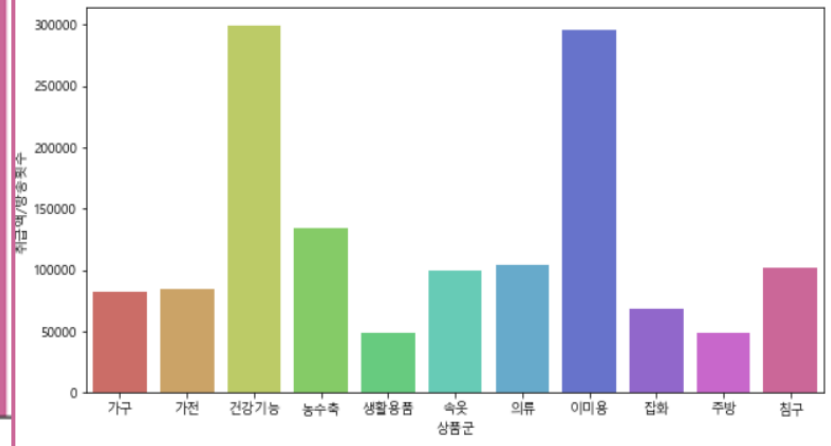
9월



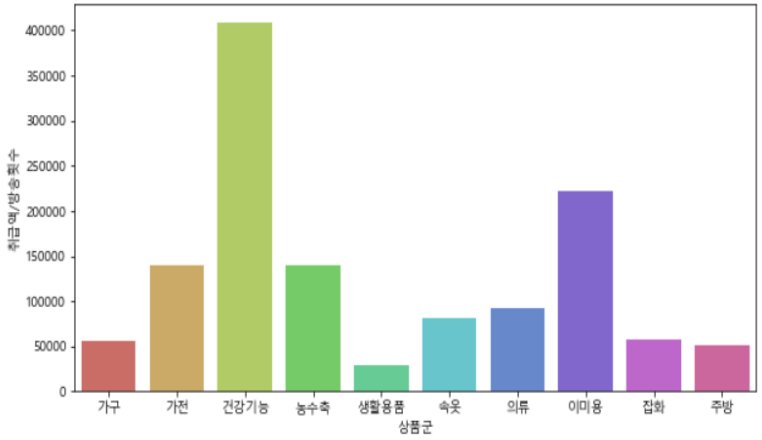
10월



11월



12월



4 데이터 분석

1월 생활용품

상위 20위 중
대부분 '비데'
∴ **동절기 비데 판매 ↑**

8월 침구류

여름용 대자리, 침구류
인기상품
∴ **여름철 침구류 판매 ↑**

10월 침구류

동절기용 카페트, 침구류
인기상품
∴ **겨울철 카페트 판매 ↑**

```
df[(df['상품군'] == '생활용품') & (df['월'] == 1)].sort_values(['취급액'], ascending = False).head()
```

	방송일시	노출(분)	마더코드	상품코드	상품명	상품군	판매단가	취급액	성별	지불	...	눈	1시간평균 미세먼지농도(㎍/㎥)	팔린제품수	기업분류	히트상품	월	일	시	히트상품
501	2019-01-06 19:40:00	20.0	100501	201514	노비타 스마트 비데	생활용품	189000	112361000	3	3	...	0	22.747555	595	Small	1	1	6	19	1
232	2019-01-03 20:40:00	20.0	100501	201514	노비타 스마트 비데	생활용품	189000	107418000	3	3	...	0	25.397671	568	Small	1	1	3	20	1
592	2019-01-07 19:40:00	20.0	100501	201514	노비타 스마트 비데	생활용품	189000	88450000	3	3	...	0	39.467801	468	Small	1	1	7	19	1
231	2019-01-03 20:20:00	20.0	100501	201514	노비타 스마트 비데	생활용품	189000	83995000	3	3	...	0	25.397671	444	Small	1	1	3	20	1
1934	2019-01-21 20:40:00	20.0	100501	201514	노비타 스마트 비데	생활용품	189000	82572000	3	3	...	0	32.756717	437	Small	0	1	21	20	0

5 rows × 34 columns

```
df[(df['상품군'] == '침구') & (df['월'] == 8)].sort_values(['취급액'], ascending = False).head()
```

	방송일시	노출(분)	마더코드	상품코드	상품명	상품군	판매단가	취급액	성별	지불	...	눈	1시간평균 미세먼지농도(㎍/㎥)	팔린제품수	기업분류	히트상품	월	일	시	히트상품
22944	2019-08-07 14:40:00	20.0	100447	201375	안지 청풍 대자리	침구	50900	31497000	3	3	...	0	8.595705	619	Small	0	8	7	14	0
22943	2019-08-07 14:20:00	20.0	100447	201375	안지 청풍 대자리	침구	50900	24605000	3	3	...	0	8.595705	483	Small	0	8	7	14	0
25037	2019-08-27 13:40:00	20.0	100200	200682	효재 화원 자수 광목 워싱 침구세트 킹	침구	149000	19352000	3	3	...	0	8.370574	130	Small	0	8	27	13	0
22942	2019-08-07 14:00:00	20.0	100447	201375	안지 청풍 대자리	침구	50900	18040000	3	3	...	0	8.595705	354	Small	0	8	7	14	0
25034	2019-08-27 13:20:00	20.0	100200	200682	효재 화원 자수 광목 워싱 침구세트 킹	침구	149000	15121000	3	3	...	0	8.370574	101	Small	0	8	27	13	0

5 rows × 34 columns

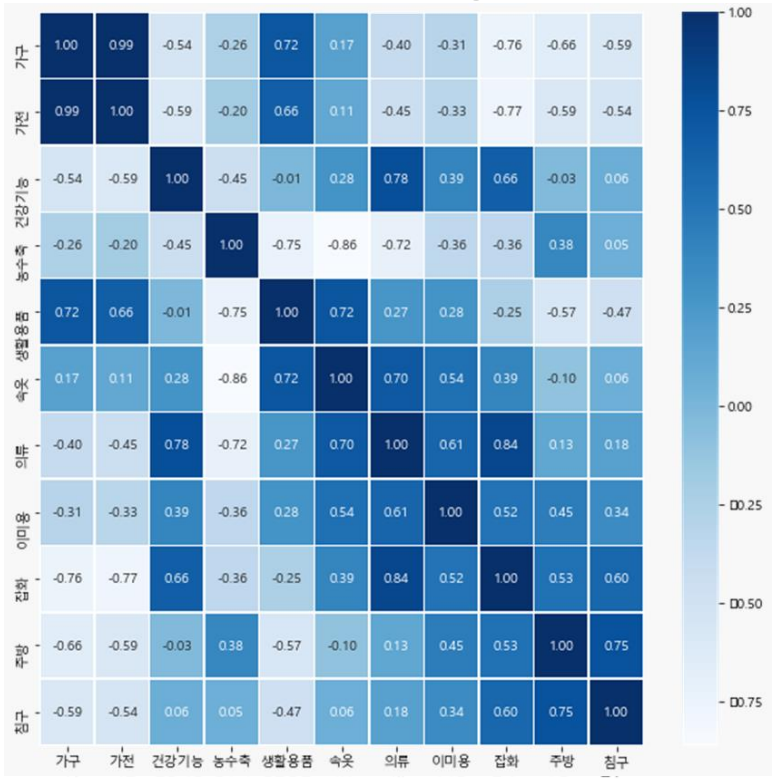
```
df[(df['상품군'] == '침구') & (df['월'] == 10)].sort_values(['취급액'], ascending = False).head()
```

	방송일시	노출(분)	마더코드	상품코드	상품명	상품군	판매단가	취급액	성별	지불	...	눈	1시간평균 미세먼지농도(㎍/㎥)	팔린제품수	기업분류	히트상품	월	일	시	히트상품
31311	2019-10-28 23:20:00	20.0	100813	202399	한빛 페르시안스타일 셔날 카페트 정보특대형	침구	109000	36508000	3	3	...	0	17.304097	335	Small	0	10	28	23	0
30785	2019-10-22 23:40:00	20.0	100813	202399	한빛 페르시안스타일 셔날 카페트 정보특대형	침구	109000	34069000	3	3	...	0	23.999021	313	Small	0	10	22	23	0
31314	2019-10-28 23:40:00	20.0	100813	202399	한빛 페르시안스타일 셔날 카페트 정보특대형	침구	109000	31780000	3	3	...	0	17.304097	292	Small	0	10	28	23	0
30782	2019-10-22 23:20:00	20.0	100813	202399	한빛 페르시안스타일 셔날 카페트 정보특대형	침구	109000	25263000	3	3	...	0	23.999021	232	Small	0	10	22	23	0
31379	2019-10-29 14:40:00	20.0	100145	200404	보풀드 헤르만 착번아웃 극세사 침구세트 SK(슈퍼킹)	침구	89900	14377000	3	3	...	0	32.407594	160	Small	0	10	29	14	0

5 rows × 34 columns

4 데이터 분석

Heat Map



시간별 상품군, 취급액 평균값을 통해
상품군 간의 상관관계 파악
→ 비슷한 시간대에 팔면 판매량 ↑ 상품군 찾기
Ex. (가구, 가전), (가구, 생활용품), (잡화, 의류)

	가구	가전	건강기능	농수축	생활용품	속옷	의류	이미용	잡화	주방	침구
가구	1.00	0.96	-0.25	-0.22	0.62	0.21	-0.16	-0.02	-0.42	-0.18	-0.20
가전	0.96	1.00	-0.27	-0.17	0.53	0.16	-0.18	-0.01	-0.38	-0.05	-0.14
건강기능	-0.25	-0.27	1.00	-0.26	0.08	0.13	0.54	0.25	0.37	-0.03	-0.01
농수축	-0.22	-0.17	-0.26	1.00	-0.53	-0.52	-0.55	-0.12	-0.31	0.29	-0.09
생활용품	0.62	0.53	0.08	-0.53	1.00	0.64	0.37	0.47	-0.08	-0.11	-0.16
속옷	0.21	0.16	0.13	-0.52	0.64	1.00	0.55	0.39	0.36	0.20	0.18
의류	-0.16	-0.18	0.54	-0.55	0.37	0.55	1.00	0.44	0.80	0.23	0.04
이미용	-0.02	-0.01	0.25	-0.12	0.47	0.39	0.44	1.00	0.32	0.51	0.28
잡화	-0.42	-0.38	0.37	-0.31	-0.08	0.36	0.80	0.32	1.00	0.44	0.44
주방	-0.18	-0.05	-0.03	0.29	-0.11	0.20	0.23	0.51	0.44	1.00	0.51
침구	-0.20	-0.14	-0.01	-0.09	-0.16	0.18	0.04	0.28	0.44	0.51	1.00

5

비즈니스

팔린제품수

```
# plot 함수를 써서 팔린제품 수를 시각화
def plot(product):
    product_hour = product['팔린제품수'].groupby([product["방송일시"].dt.hour]).mean()
    product_hour.sort_values(inplace=True, ascending = False)
    product_hour.plot(kind='bar', y=['팔린제품수'], figsize=(16,8), fontsize=15, alpha=0.7, stacked=True)
    plt.xlabel('시간', fontsize=12)
    plt.xticks(rotation=0)
    plt.ylabel('팔린제품 수', fontsize=12)
    plt.title('NS 홈쇼핑 팔린제품 수', fontsize=20)
    plt.legend(fontsize=10)

# Prime Time 등급을 판단하고 프라임 시간대 Column을 추가하는 것
def grade(product):
    product_hour = product['팔린제품수'].groupby([product["방송일시"].dt.hour]).mean()
    product_hour.sort_values(inplace=True, ascending = False)
    mean = (product_hour / sum(product_hour)).mean()
    product_hour['프라임 시간대'] = np.where((product_hour / sum(product_hour)) > mean*1.5,'Prime','General')
    return product_hour['프라임 시간대']
```

팔린제품 수 시각화

STEP 1

Prime Time 판별 기준

$$\text{mean} = \frac{\text{각 시간별로 팔린제품 수}}{\text{각 시간별로 팔린제품 수의 총합}}$$

$$\frac{\text{상품군에 따른 각 시간별로 팔린제품}}{\text{상품군에 따른 각 시간별로 팔린제품 수의 총합}} > 1.5 \times \text{mean}$$

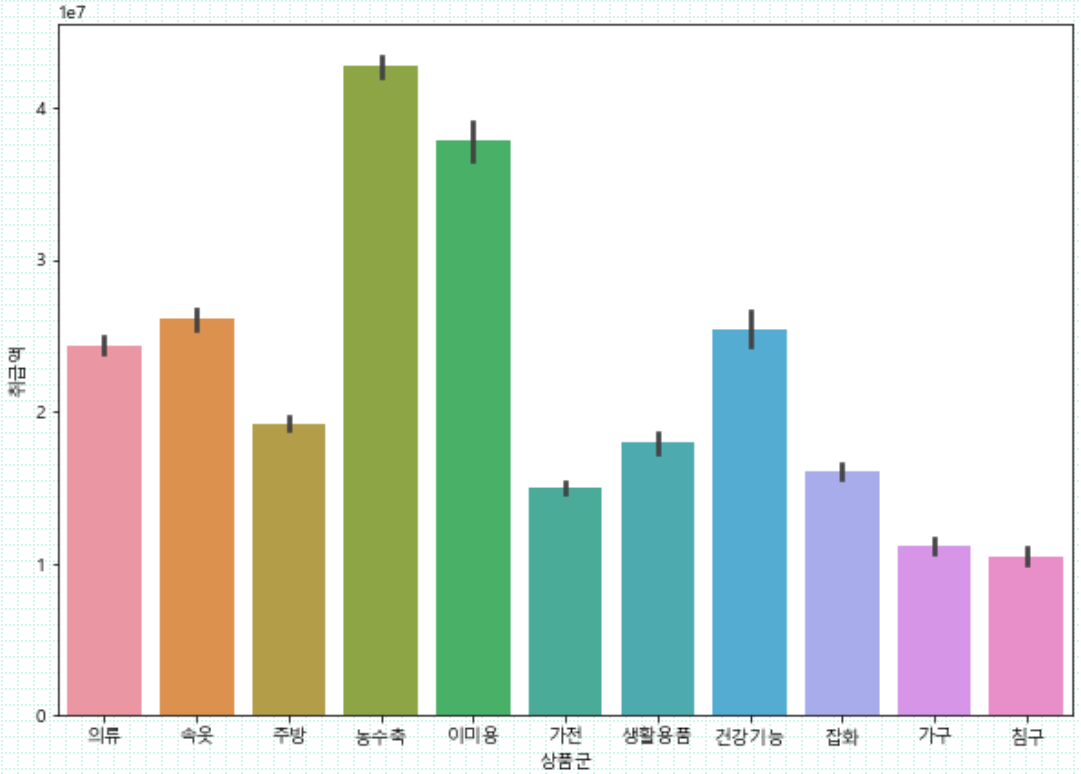
일 경우

Prime Time 취급

Prime Time 설정

5 비즈니스

취급액



상품군별 취급액을 나타낸 그래프 / X축: 상품군, Y축: 취급액

STEP 2

취급액이 높은 순서 배열

Prime Time이 겹칠 경우
우선 순위를 두기 위하여

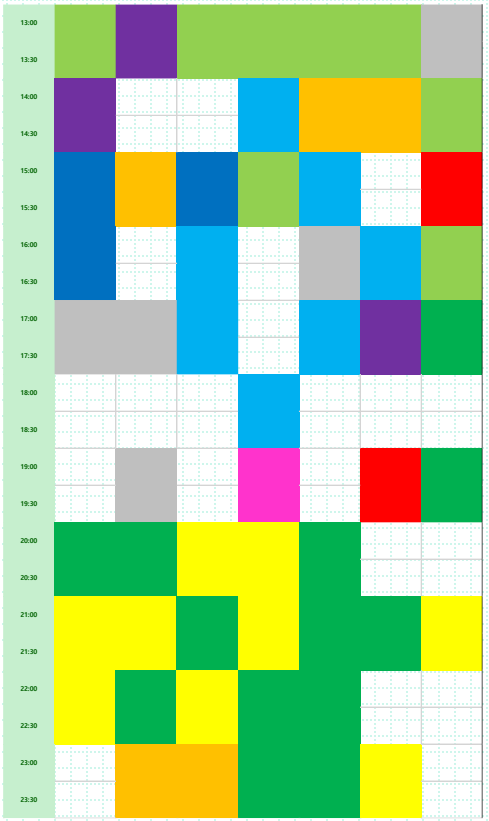
취급액 순위 시각화

Seaborn Library 사용

5

비즈니스

엑셀



가구 | 가전 | 건강기능 | 농수축 | 생활용품 | 이미용 | 잡화 | 주방 | 침구 | 속옷 | 의류 | Total

STEP 3

요일별, 시간대별

Prime Time을 쉽게
파악하기 위하여

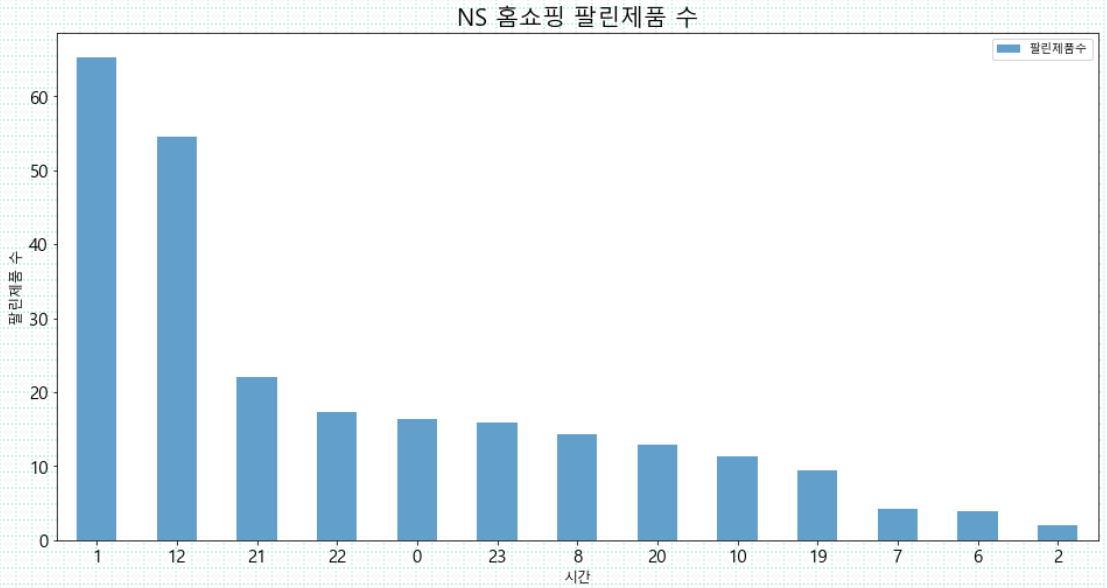
엑셀로 시각화

Excel 사용

5

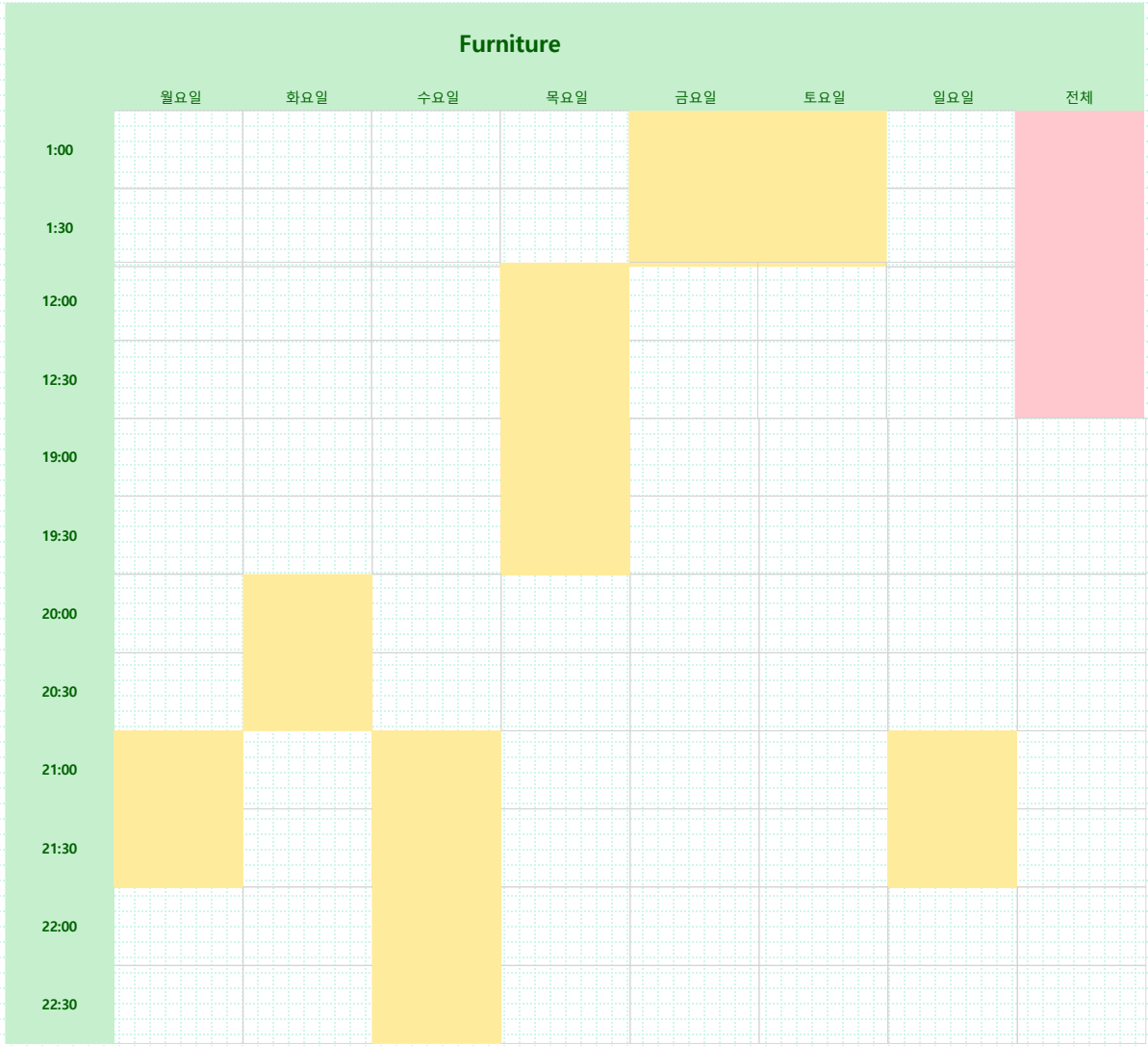
비즈니스

가구



가구 Prime Time

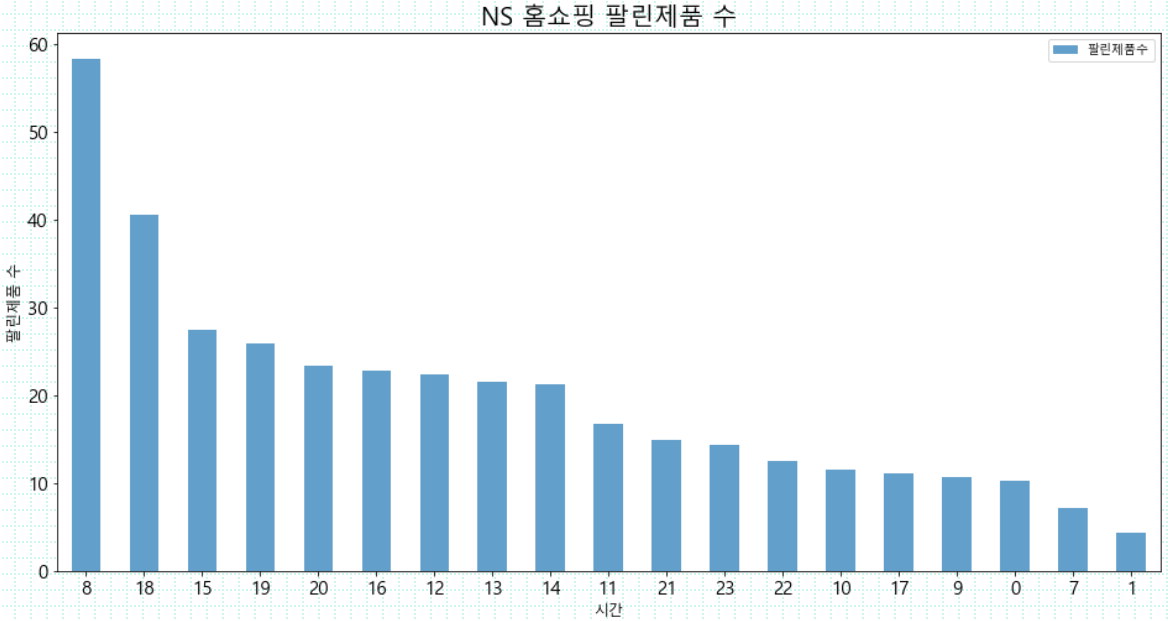
월요일	화요일	수요일	목요일	금요일	토요일	일요일
21시	20시	21시~22시	12시, 19시	1시	1시	21시



5

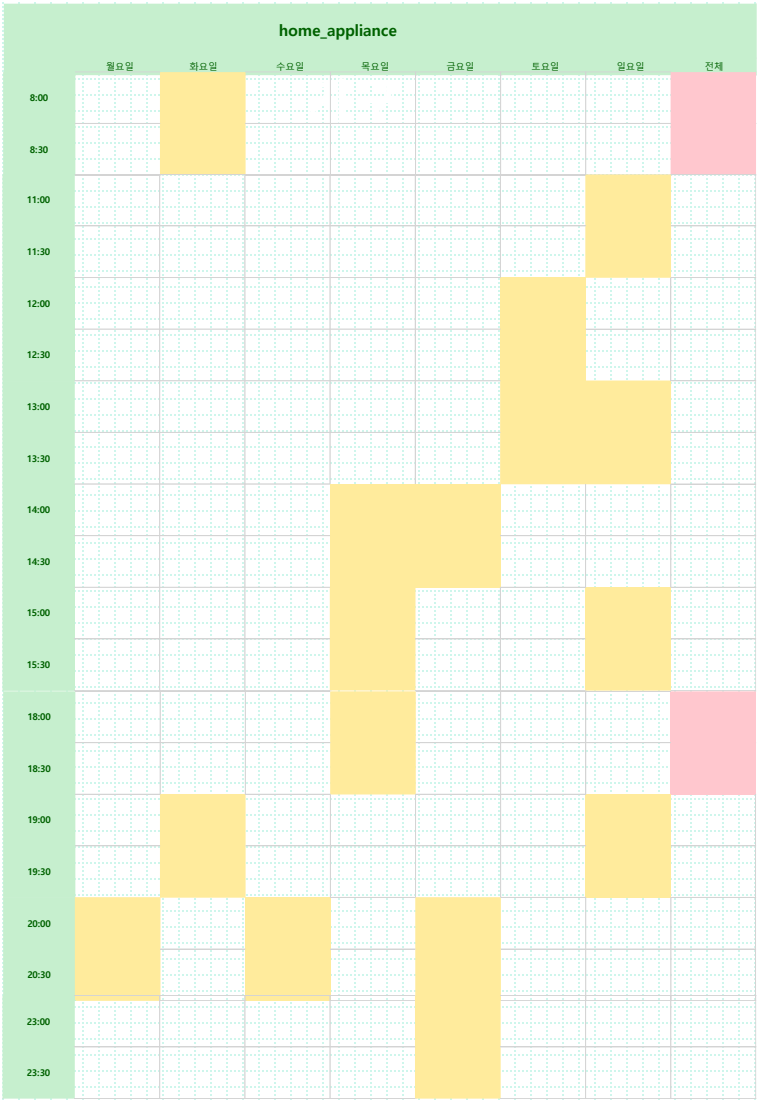
비즈니스

가전



가전 Prime Time

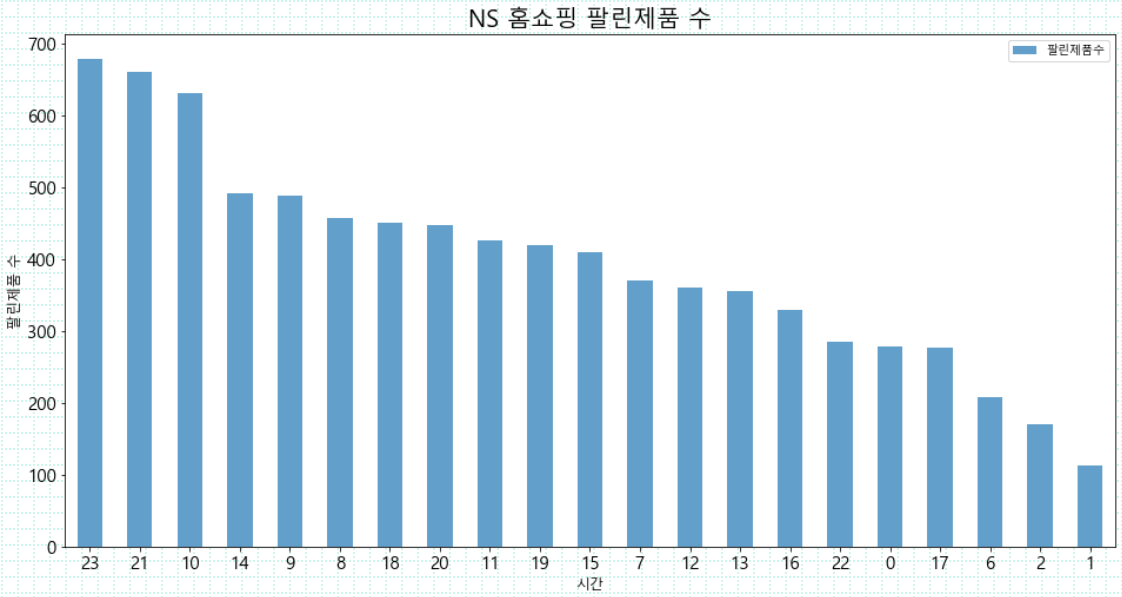
월요일	화요일	수요일	목요일	금요일	토요일	일요일
20시	8시, 17시	20시	14시, 15시 18시	14시,20시 21시	12시, 13시	11시, 13시 15시, 19시

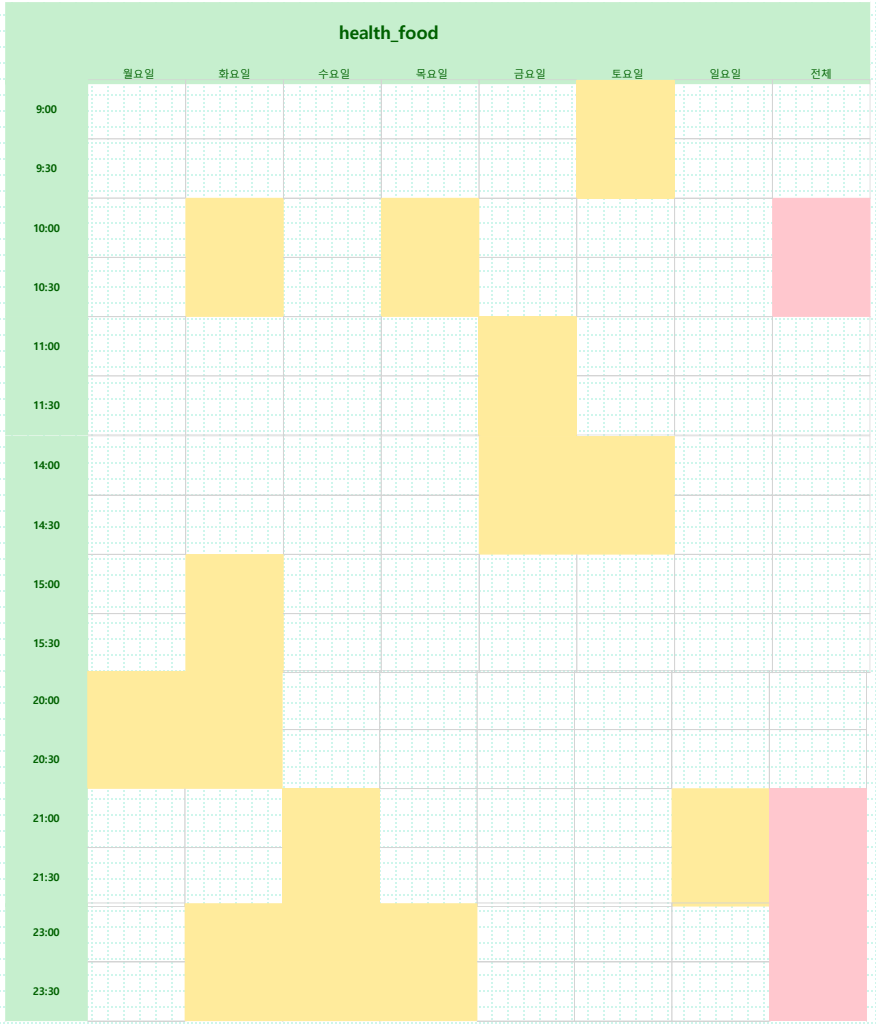


5

비즈니스

건강기능



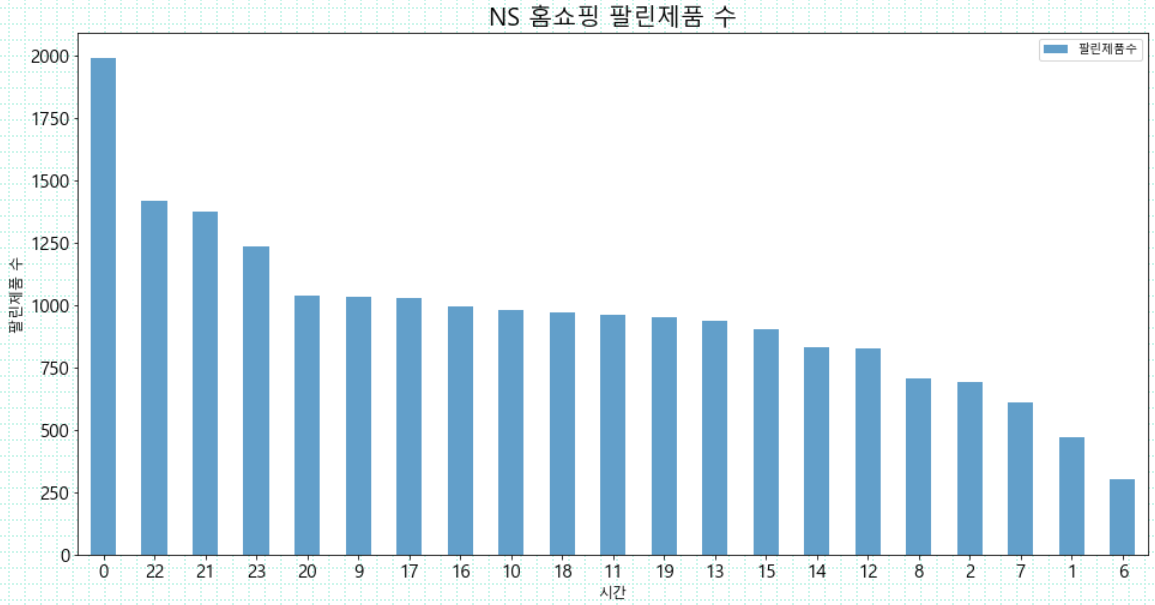


건강기능 Prime Time

월요일	화요일	수요일	목요일	금요일	토요일	일요일
20시	10시, 15시, 20시	21시~22시	10시, 23시	11시, 14시	9시, 14시	21시

비즈니스

농수축

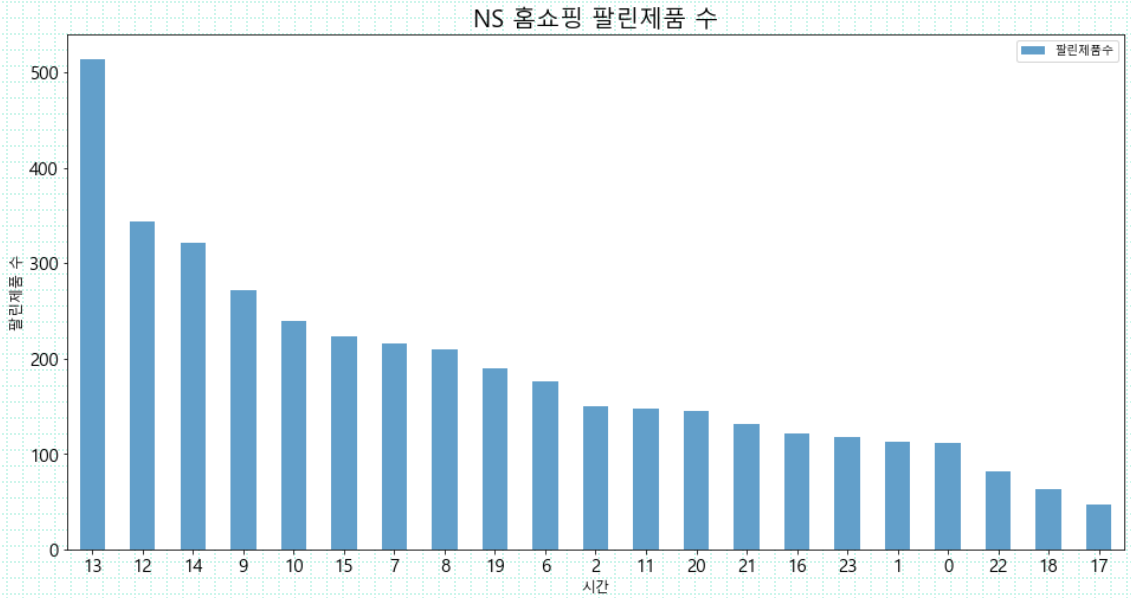


농수축 Prime Time

월요일	화요일	수요일	목요일	금요일	토요일	일요일
0시, 21시, 22시	21시	20시, 22시	12시, 20시, 21시	0시	23시	21시

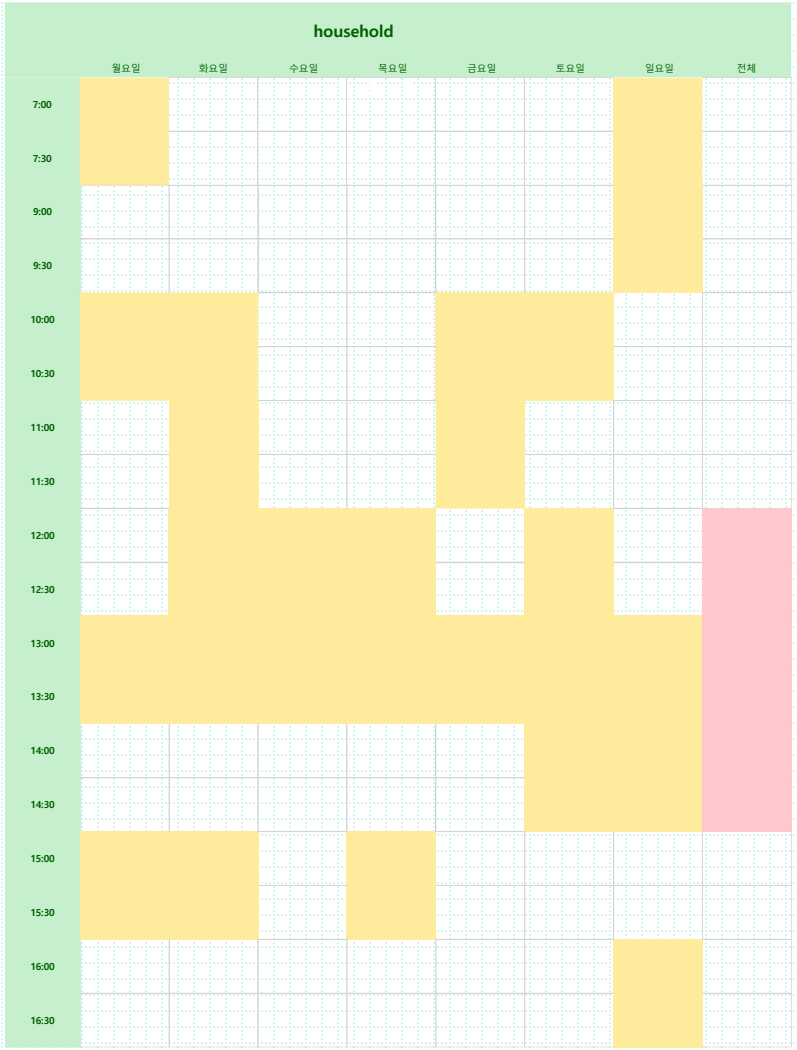
5 비즈니스

생활용품



생활용품 Prime Time

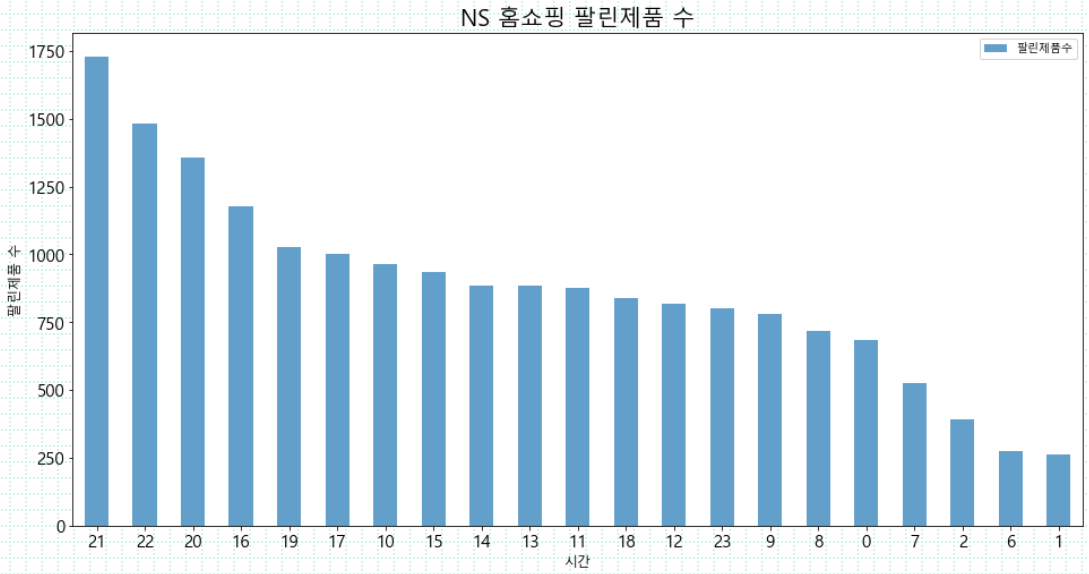
월요일	화요일	수요일	목요일	금요일	토요일	일요일
7시, 10시 13시, 15시	10시~13시, 15시	12시~13시	12시~13시, 15시	10시, 11시, 13시	10시, 12시~14시	7시, 9시, 13시, 14시, 16시



5

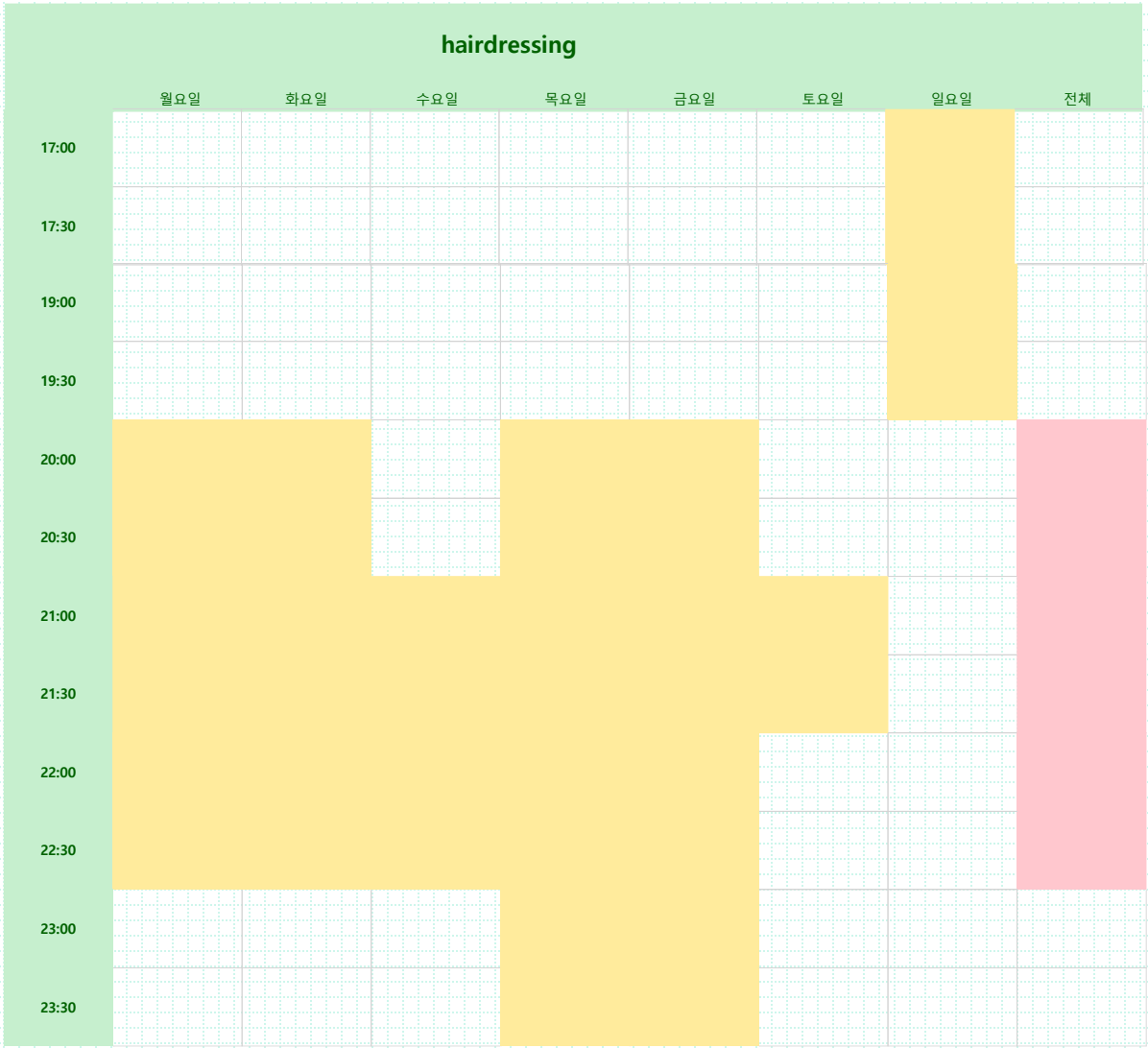
비즈니스

이미용



이미용 Prime Time

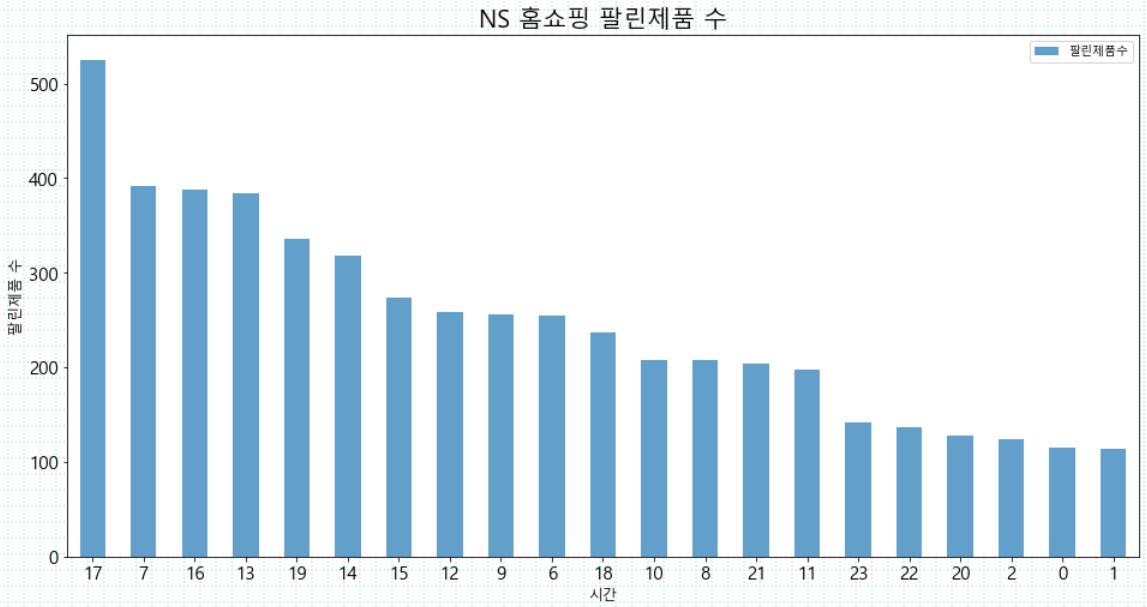
월요일	화요일	수요일	목요일	금요일	토요일	일요일
20시~22시	20시~22시	21시~22시	20시~23시	20시~23시	21시	17시, 19시



5

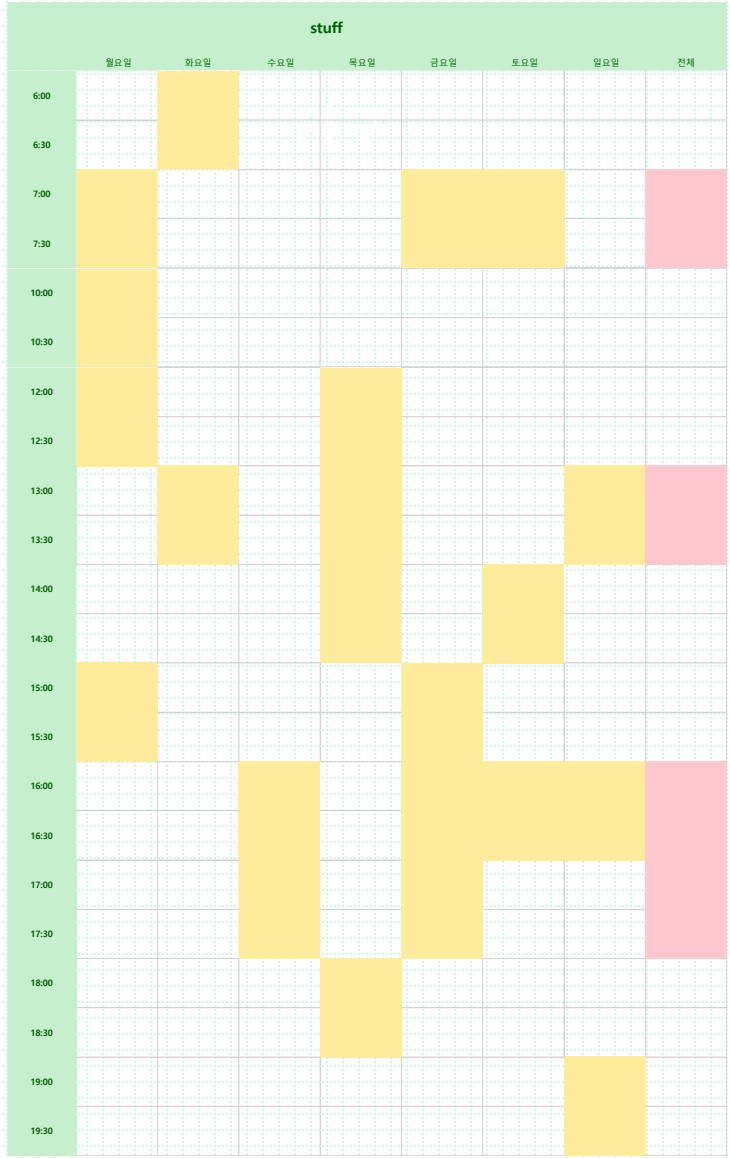
비즈니스

잡화



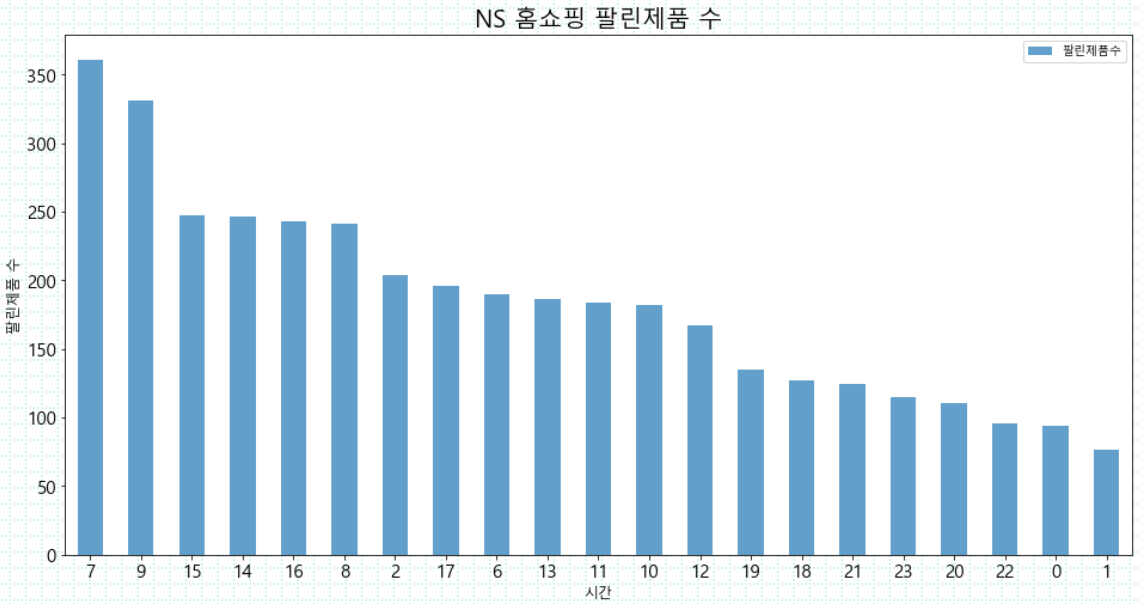
잡화 Prime Time

월요일	화요일	수요일	목요일	금요일	토요일	일요일
7시, 10시, 12시, 15시	6시, 13시	16시~17시	12시~14시 18시	7시, 15시 16시~17시	7시, 14시 16시	13시, 16시 19시



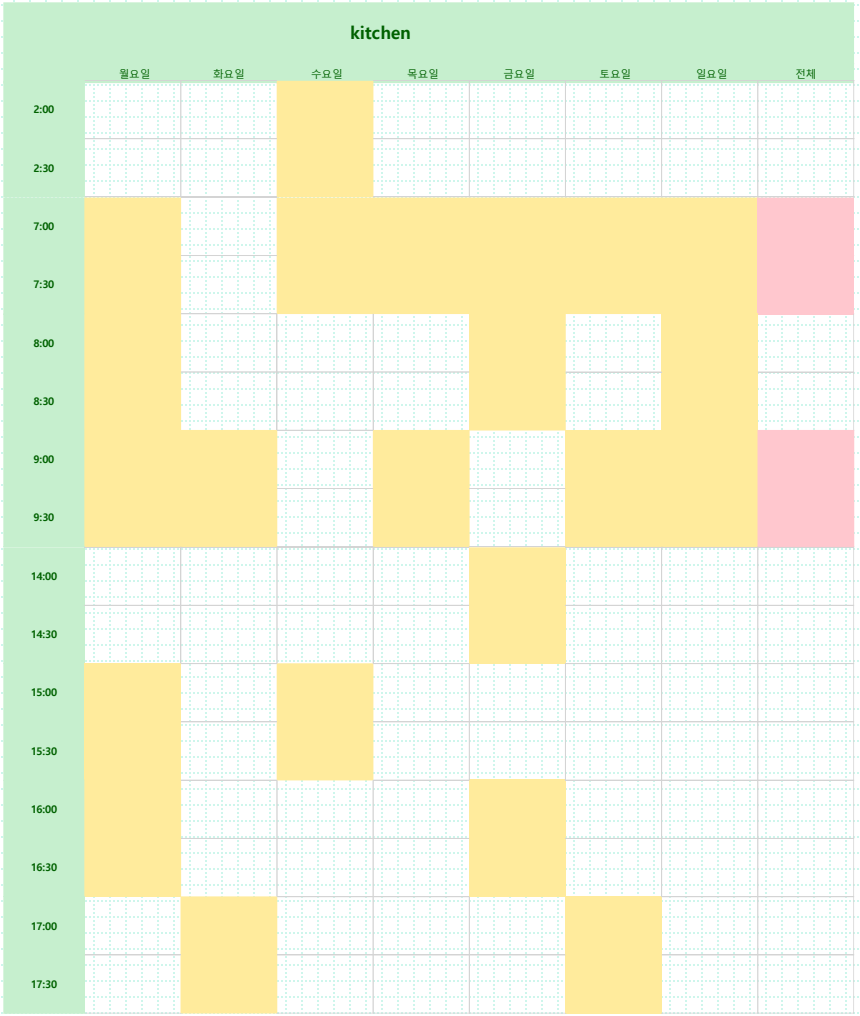
비즈니스

주방



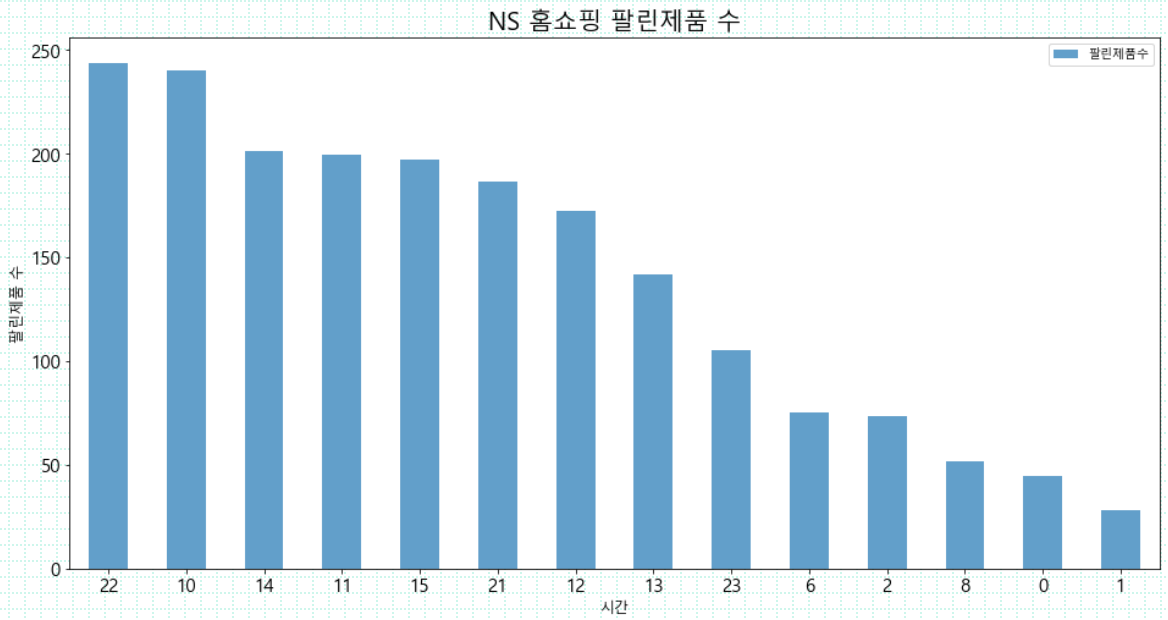
주방 Prime Time

월요일	화요일	수요일	목요일	금요일	토요일	일요일
7시~9시 15시, 16시	9시, 17시	2시, 7시 15시	7시, 9시	7시, 8시 14시, 16시	7시, 9시 17시	7시~9시



5 비즈니스

침구

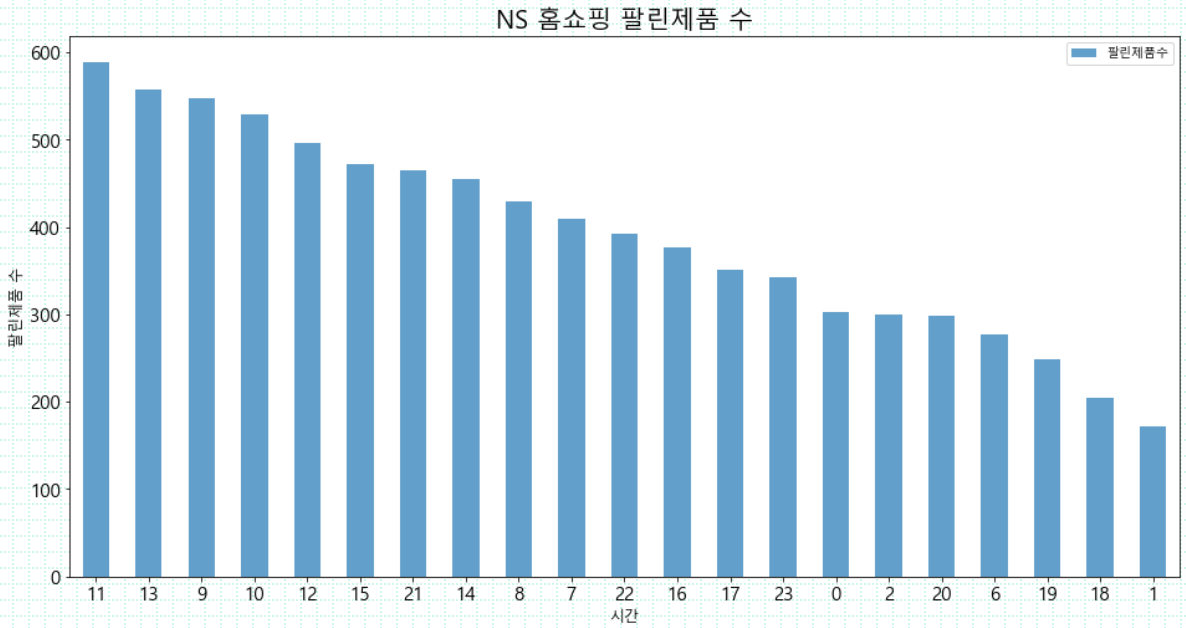


침구 Prime Time

월요일	화요일	수요일	목요일	금요일	토요일	일요일
	12시	11시, 15시	13시		13시	

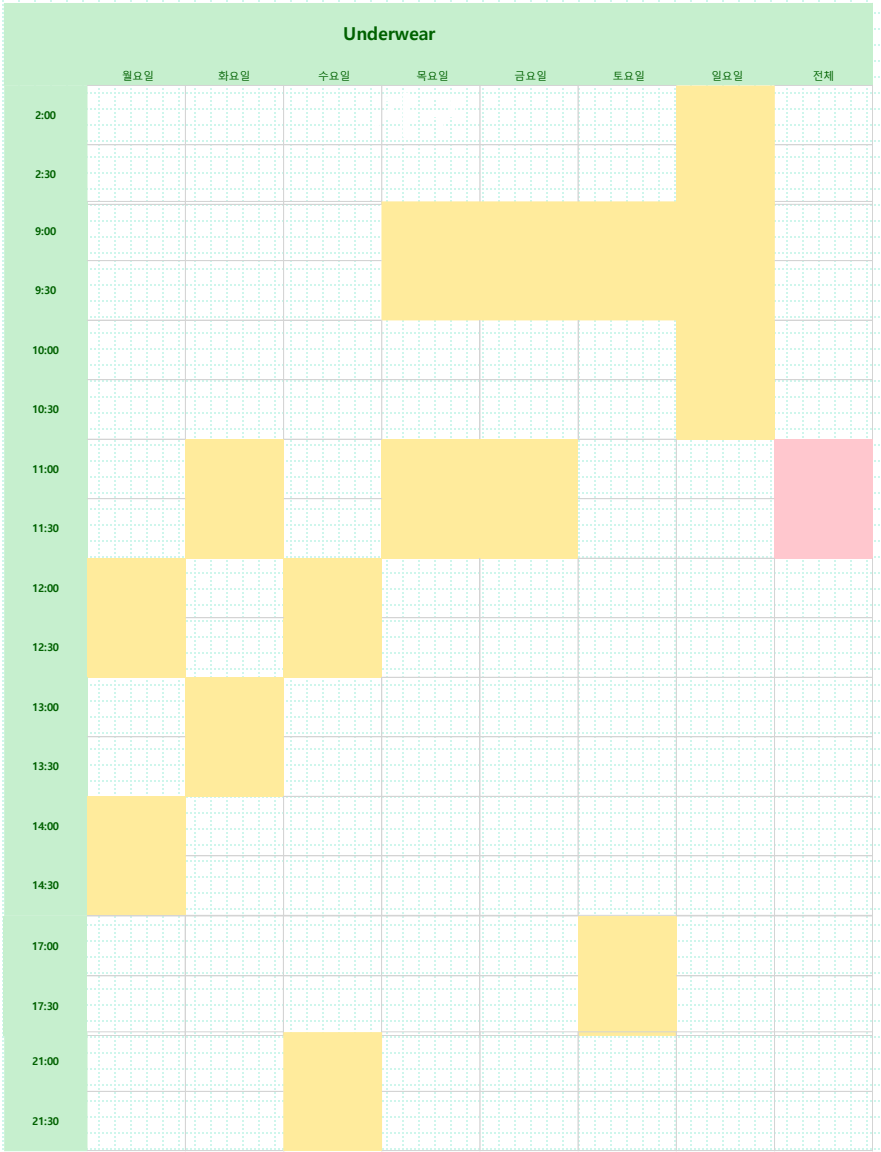
5 비즈니스

속옷



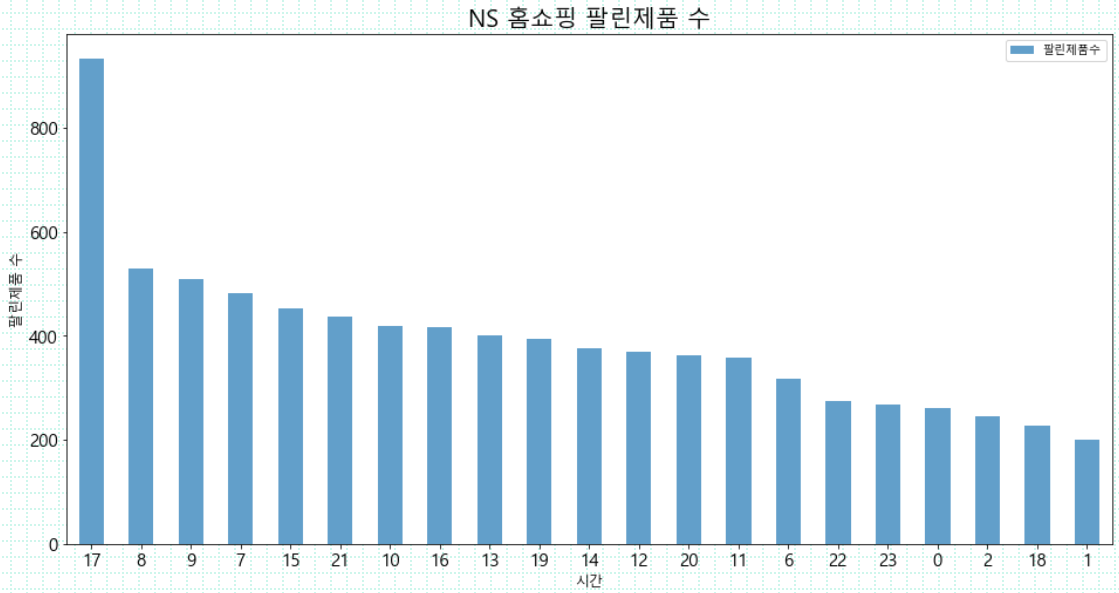
속옷 Prime Time

월요일	화요일	수요일	목요일	금요일	토요일	일요일
12시, 14시	11시, 13시	12시, 21시	9시, 11시	9시, 11시	9시, 17시	2시, 9시, 10시



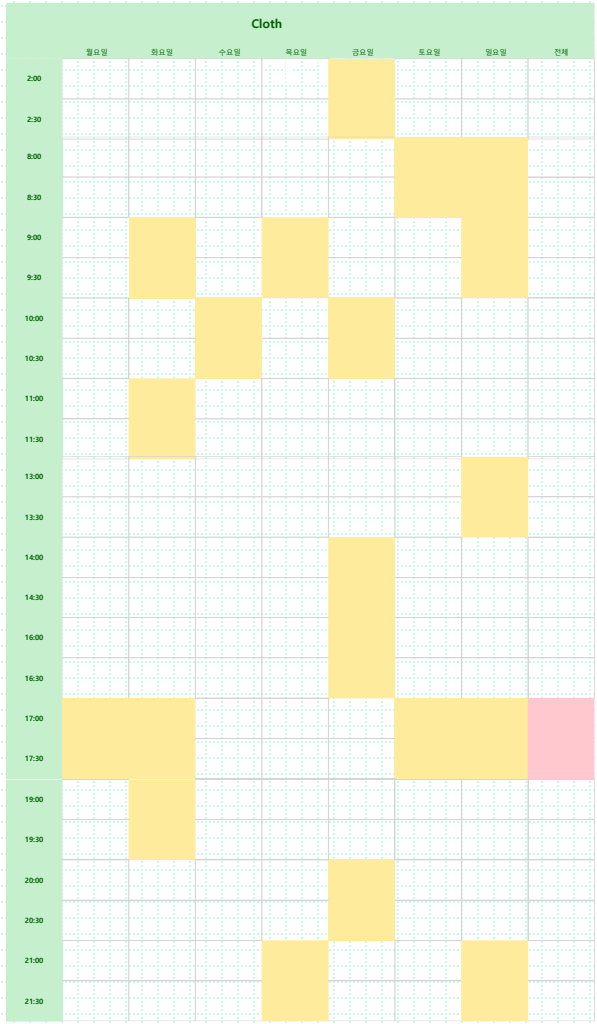
5 비즈니스

의류



의류 Prime Time

월요일	화요일	수요일	목요일	금요일	토요일	일요일
17시	9시, 11시 17시, 19시	10시	9시, 21시	2시, 10시, 14시, 16시, 20시	8시, 17시	8시, 9시, 13시, 17시, 21시

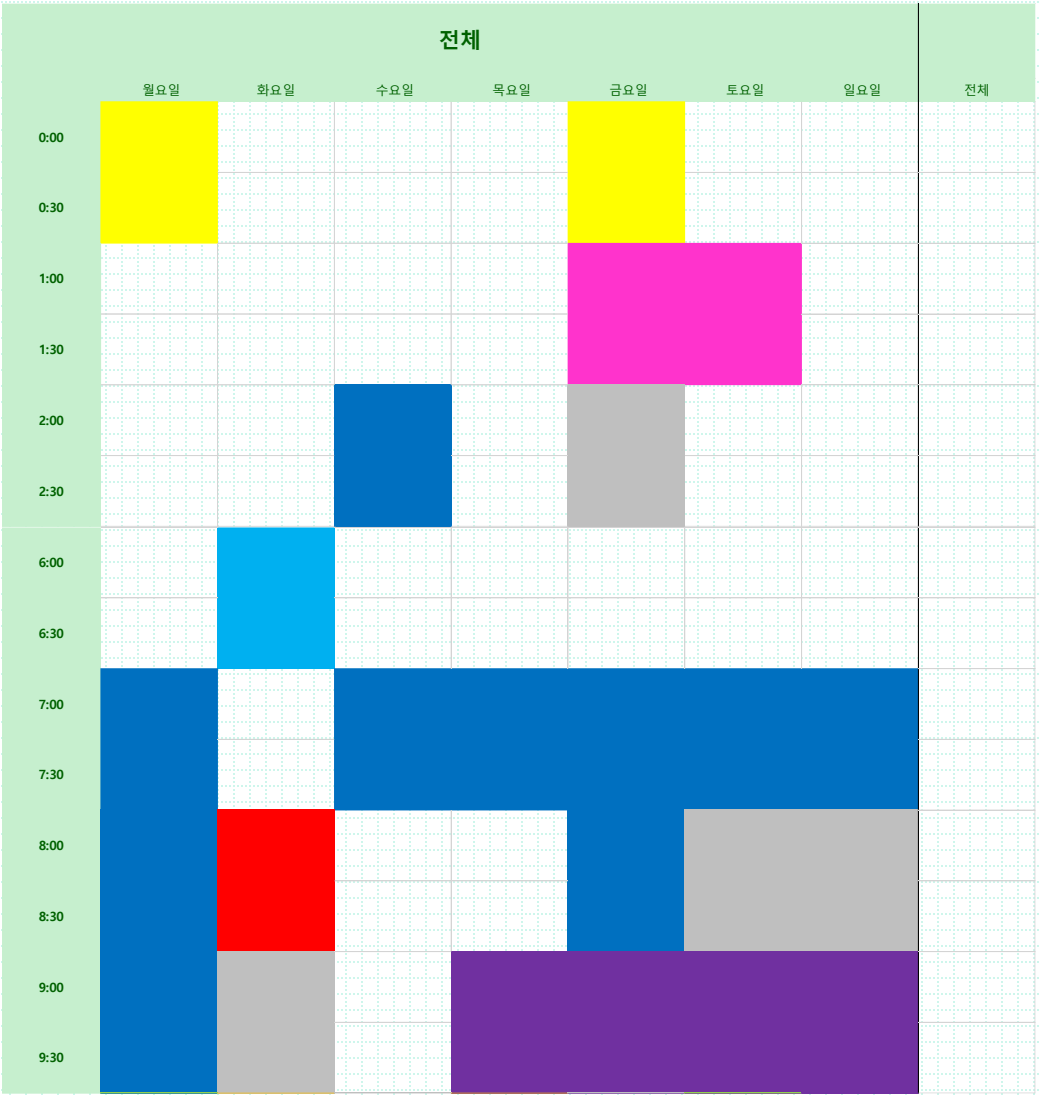


비즈니스

전체

전체 Prime Time (0시 ~ 10시)

	월	화	수	목	금	토	일
상품군	농수축 주방	잡화 가전 의류	주방	주방 속옷	농수축 가구 의류 주방 속옷	가구 주방 의류 속옷	주방 의류 속옥



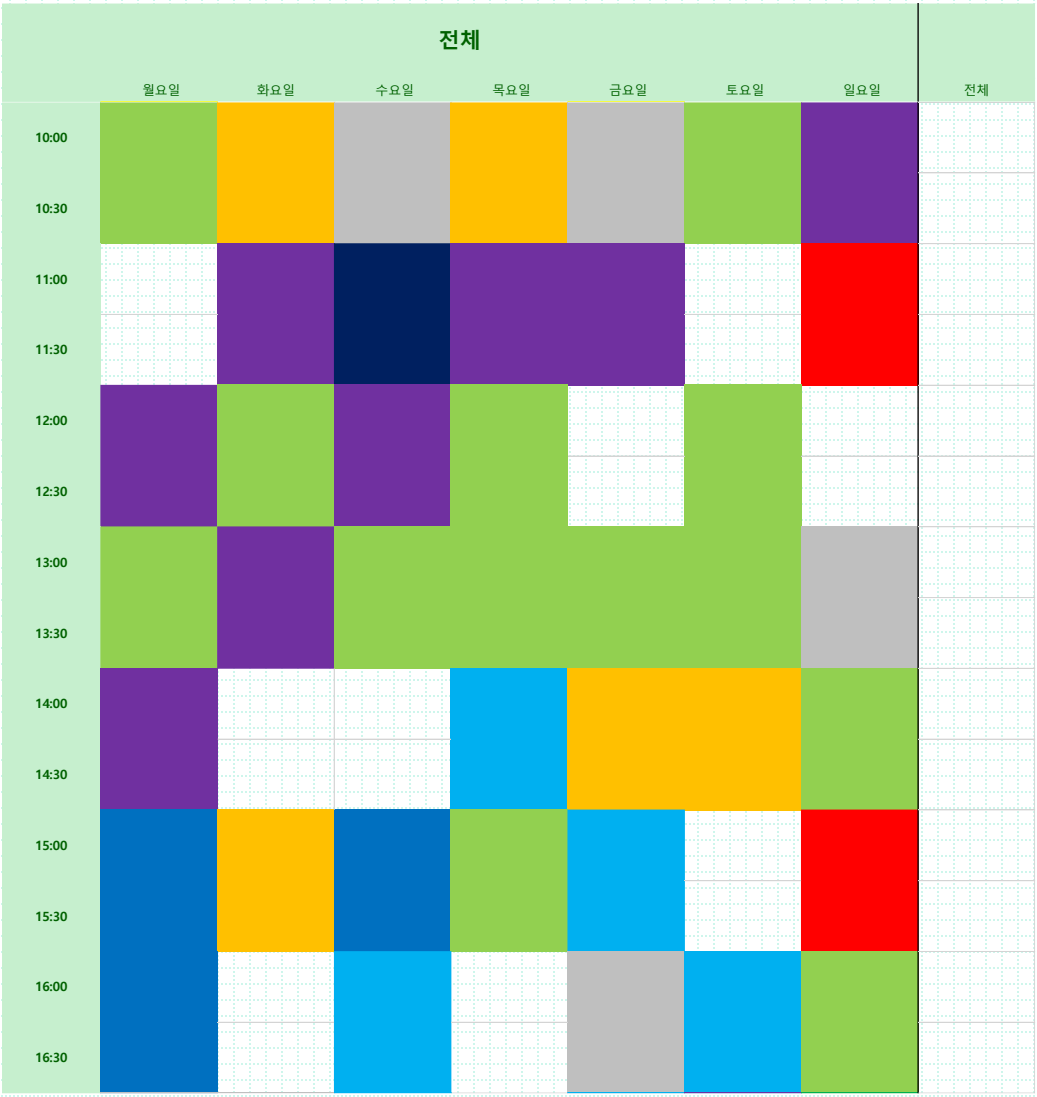
가구 | 가전 | 건강기능 | 농수축 | 생활용품 | 이미용 | 잡화 | 주방 | 침구 | 속옷 | 의류 | Total

비즈니스

전체

전체 Prime Time (10시 ~ 17시)

	월	화	수	목	금	토	일
상품군	생활용품 속옷 주방	건강기능 속옷 생활용품	의류 침구 속옷 생활용품 주방 잡화	건강기능 속옷 생활용품 잡화	의류 속옷 생활용품 건강기능 잡화 의류	생활용품 건강기능 잡화	속옷 가전 의류 생활용품



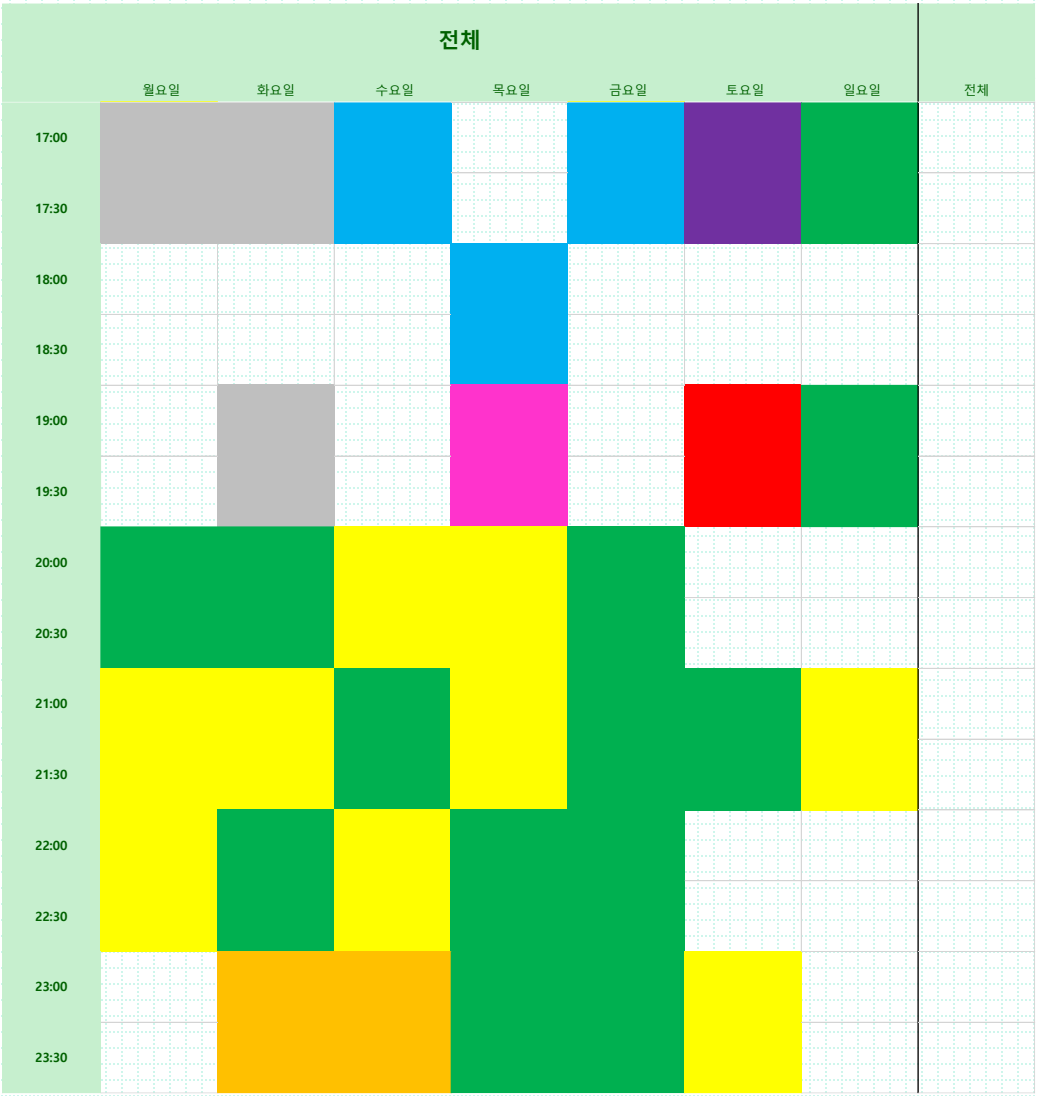
가구 | 가전 | 건강기능 | 농수축 | 생활용품 | 이미용 | 잡화 | 주방 | 침구 | 속옷 | 의류 | Total

비즈니스

전체

전체 Prime Time (17시 ~ 0시)

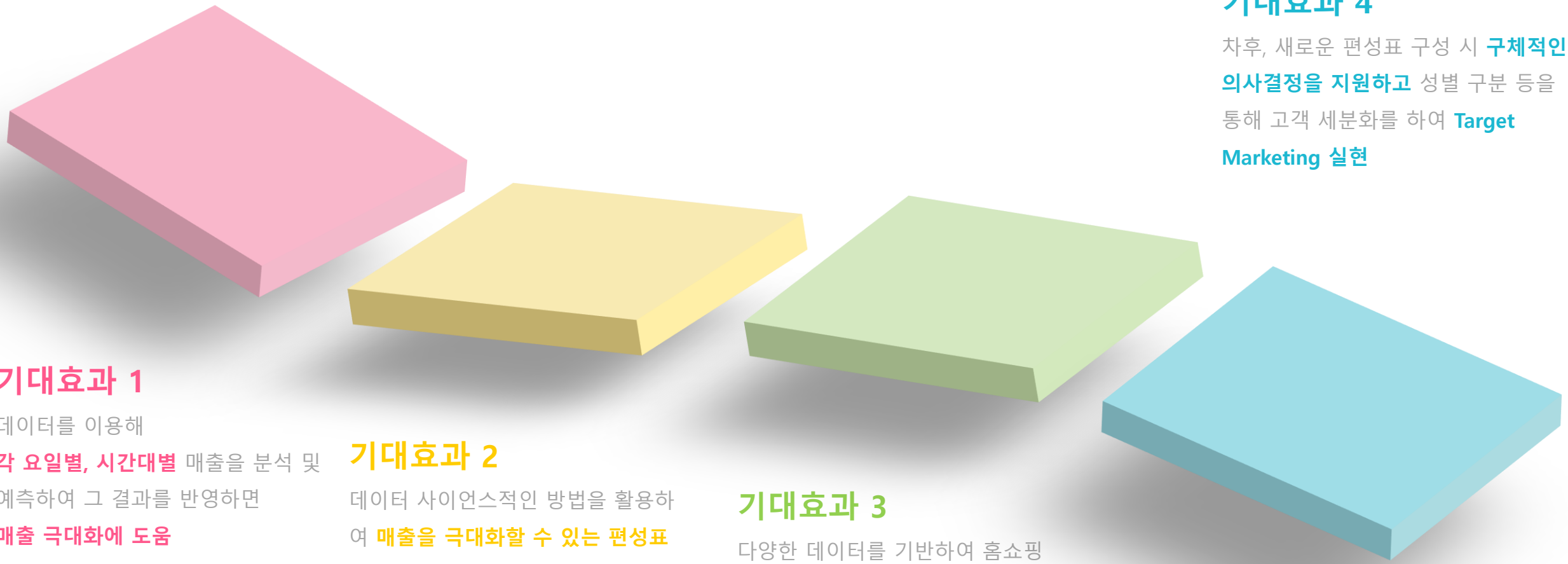
	월	화	수	목	금	토	일
상품군	의류 이미용 농수축	의류 이미용 농수축 건강기능	잡화 농수축 이미용 건강기능	잡화 가구 농수축 이미용	잡화 이미용	속옷 가전 이미용 농수축	이미용 농수축



가구 | 가전 | 건강기능 | 농수축 | 생활용품 | 이미용 | 잡화 | 주방 | 침구 | 속옷 | 의류 | Total

6

기대효과



기대효과 1

데이터를 이용해
각 요일별, 시간대별 매출을 분석 및 예측하여 그 결과를 반영하면
매출 극대화에 도움

기대효과 2

데이터 사이언스적인 방법을 활용하여 **매출을 극대화할 수 있는 편성표 구성**

기대효과 3

다양한 데이터를 기반하여 홈쇼핑 방송을 기획하고 편성표를 구성하였으므로 **운영비용 절감 기여**

기대효과 4

차후, 새로운 편성표 구성 시 **구체적인 의사결정을 지원하고** 성별 구분 등을 통해 고객 세분화를 하여 **Target Marketing 실현**