In [37]:

```
# 기본 라이브러리 호출
import warnings
warnings.filterwarnings('ignore')
warnings.warn("once")
import pandas as pd
import numpy as np
import scipy as sp
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from scipy import stats
from scipy.stats import norm, skew
# 선형모델을 추정하는 라이브러리
import statsmodels.formula.api as smf
import statsmodels.api as sm
import statsmodels.stats.api as sms
from patsy import dmatrices
color = sns.color_palette()

pd.set_option('display.float_format','{:,.2f}'.format) # 소수점 2번째 자리까지 표현
pd.set_option('display.max_columns', None) # 모든 컬럼 표시
pd.set_option('display.max_colwidth', None) # 컬럼내용 전체 표시

#Graph에 한글을 표시하기 위한 코드
import matplotlib
from matplotlib import font_manager, rc
import platform
# font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
font_name = font_manager.FontProperties(fname="/usr/share/fonts/NanumGothicCoding.ttf").get_nam
rc('font', family=font_name)\

matplotlib.rcParams['axes.unicode_minus'] = False
```

- 실제 서비스 내 고객 데이터 활용 과정을 실무적인 관점에서 설명해주는 사이트에서 가져왔습니다. 상세 분석과정 또한 함께 나와있으나 같이 다양한 방법으로 분석해보면 좋을 듯 합니다 :)
- 주제 : 서비스 유저 행동 분석을 통해 결제 전환율을 높이기 위한 방안을 탐색

- actiontype : 유저 행동 (OPEN/CLOSE/SAVE/RESET/EXPORT)
- ismydoc : View/NoView
- ext : 출력한 문서 종류
- sessionid : 유저별 session id
- documentposition : 접속한 위치
- datetime : 해당 action이 일어난 시각
- screen : action이 일어난 페이지

- Funnel 방식으로 유저의 행동 패턴을 파악하여 해당 시나리오의 Bottleneck 구간을 탐색하자
- 유저를 세분화하여 Bottleneck 개선 방안에 유용한 Insight를 도출하자

서비스 건강도에 대한 MECE 접근법 예시

- 예를 들어, 특정 서비스의 건강도를 측정하는 지표를 계획한다고 가정해보자. 서비스 건강도를 일별 방문자수, 재방문율, 구매율 및 재구매율 등으로 서로 독립적인 하위 지표로 구성할 수 있으며(ME), 누락없이(CE) 측정할 수 있을 것이다.
- 서비스 건강도
  - 방문자수
    - 재방문자수 및 재방문율
    - 신규방문자수
  - 이용자수
    - 특정 기능 이용자수
      - 신규유저
      - 기존유저
    - 특정 기능 재이용자수 (재이용율)
      - 신규유저
      - 기존유저
  - 구매자수
    - 재구매자 및 재구매율
    - 신규구매자 및 구매전환율
    - ARPU, ARPPU, LTV
  - 추천(공유)수
    - 추천 제공자수
    - 추천으로 인해 유입된 유저수

1.1 결측치 처리 및 명목형 변수 전처리 하시오

In [38]:

```
df = pd.read_csv('./data/df_funnel.csv', index_col = 0)
```

In [39]:

```
print(df.shape)
df.head()
```

...

In [40]:

```
df['actiontype'].value_counts()
```

...

In [41]:

```
df['ismydoc'].value_counts()
```

...

In [42]:

```
df['ext'].value_counts()
```

...

In [43]:

```
df['documentposition'].value_counts()
```

...

In [44]:

```
df['screen'].value_counts()
```

...

In [45]:

```
df.isna().sum()
```

...

In [46]:

```
df['datetime'] = pd.to_datetime(df['datetime'])
```

In [47]:

```
# dict 생성
ext_dic = {'DOCX' : 'DOC',
           'XLSX' : 'XLS',
           'PPTX' : 'PPT',
           'PPSX' : 'PPT',
           'PPS' : 'PPT',
           'ODT' : 'TXT',
           'PNG' : 'JPG',
           'WORD' : 'DOC',
           'SHEET' : 'XLS'}

# 해당 컬럼에 replace 함수와 dict 적용
df['ext'] = df['ext'].replace(ext_dic)
```

In [48]:

```
df['ext'].value_counts()
```

Out[48]:

```
DOC    82891
PDF    82004
XLS    76612
HWP    26244
PPT    23465
TXT    10634
JPG       11
Name: ext, dtype: int64
```

In [49]:

```
act_dic = {'SAVEAS': 'SAVE',
           'SAVEAS_OTHER': 'SAVE',
           'EXPORT_SAME': 'EXPORT'
            }

df['actiontype'] = df['actiontype'].replace(act_dic)

```

In [50]:

```
df['ext'].value_counts()
```

...

In [51]:

```
df['actiontype'].value_counts()
```

...

In [52]:

```
df['sessionid'].duplicated().sum()
```

Out[52]:

```
186867
```

In [53]:

```
cols = df.columns # list of columns' names

# loop
for c in cols:
    if c != 'datetime':
        df[c] = df[c].apply(lambda x: x.lower())
```

In [54]:

```
s = [] # empty list
j = 0 # default setting

# loop
for i in range(len(df)-1):

    # compare each rows
    if df.loc[i, 'sessionid'] == df.loc[i+1, 'sessionid']:
        s.append(j)

    # update j values
    else:
        s.append(j)
        j += 1
```

In [55]:

```
s[:20]
```

Out[55]:

```
[0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 6, 7, 7, 7]
```

In [56]:

```
df['sessionid']  = pd.Series(['sess' + str(x) for x in s]) # convert to string as add character
```

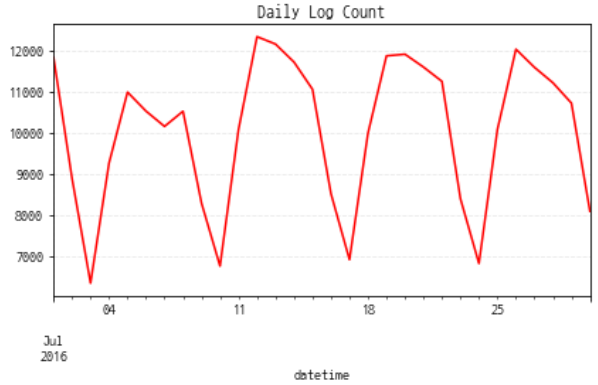
1.2 EDA를 통해 데이터 시각화를 진행하시오

1.2.1 일별 Trend

- 탐색 내용
  - 일별 로그 카운트
  - 일별 세션 카운트
  - 요일별 세션 카운트

In [57]:

```
# daily log size
df.groupby('datetime').size().plot(c = 'r')
plt.title('Daily Log Count')
plt.grid(color = 'lightgrey', alpha = 0.5, linestyle = '--')
plt.tight_layout()
```



In [58]:

```
# daily session count => activness index
df.groupby('datetime')['sessionid'].nunique().plot(c= 'b')
plt.title('Daily Session')
plt.grid(color = 'lightgrey', alpha = 0.5, linestyle = '--')
plt.tight_layout()
```

...

Note.

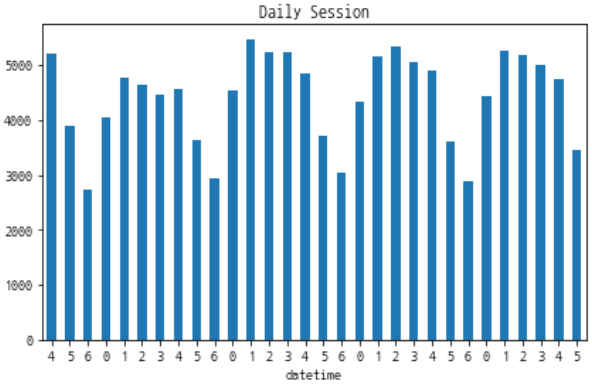
- 앱 사용에 seasonality 존재
- 로그수와 세션수의 트렌드가 유사

In [59]:

```
size = df.groupby('datetime').size()
```

In [60]:

```
s = df.groupby('datetime')['sessionid'].nunique()
s.index = s.index.dayofweek

s.plot(kind = 'bar', rot = 0)
plt.title('Daily Session')
plt.tight_layout()
plt.show()
```



Note.

- 주말에 사용성이 매우 감소하고 주중 초반이 높은편
- 문서앱이라는 특성상, 직장인 혹은 학생이 주로 사용할 것으로 가정하면 당연한 결과

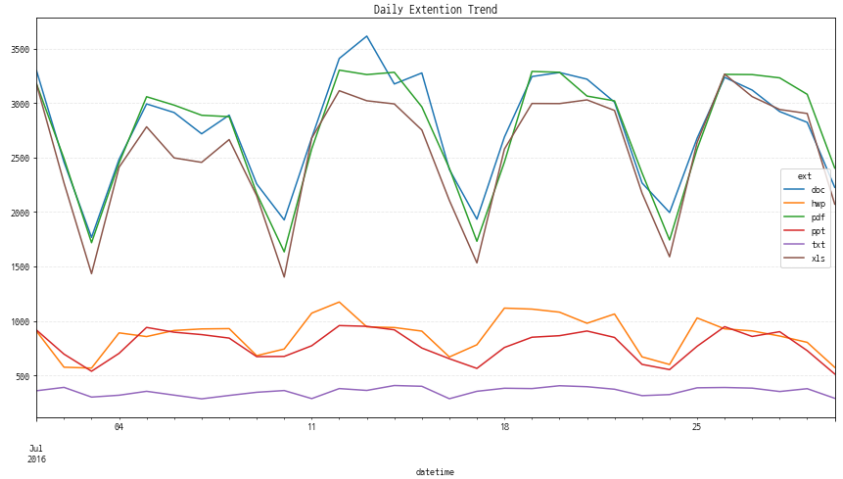
- 탐색 내용
  - 일별, 확장자별 로그수
  - 일별, 위치별 로그수
  - 일별, 액션별 로그수
  - 일별, 화면 스크린별 유니크 유저수

In [62]:

```
# 일별, 카테고리별 카운트 (피벗팅)
p = df.groupby(["datetime", "ext"]).size().unstack().dropna(axis=1) # unstack으로 재구조화
```

In [67]:

```
# daily trend by extention
df.groupby(["datetime", "ext"]).size().unstack().dropna(axis=1).plot(figsize=(12,7));

plt.title("Daily Extention Trend")
plt.grid(color='lightgrey', alpha=0.5, linestyle='--')
plt.tight_layout()
```



In [71]:

```
# daily trend by doc position
df.groupby(["datetime", "documentposition"]).size().unstack().dropna(axis=1).plot(figsize=(12,7
3
4 plt.title("Daily documentposition")
5 plt.grid(color='lightgrey', alpha=0.5, linestyle='--')
6 plt.tight_layout()
```

. . .

In [72]:

```
# daily trend by action type
df.groupby(["datetime", "actiontype"]).size().unstack().dropna(axis=1).plot(figsize=(12,7));

plt.title("Daily actiontype")
plt.grid(color='lightgrey', alpha=0.5, linestyle='--')
plt.tight_layout()
```

. . .

In [73]:

```
# daily trend by screen name
df.groupby(["datetime", "screen"]).size().unstack().fillna(0).astype(int).plot(figsize=(12,7));

plt.title("Daily screens")
plt.grid(color='lightgrey', alpha=0.5, linestyle='--')
plt.tight_layout()
```

. . .

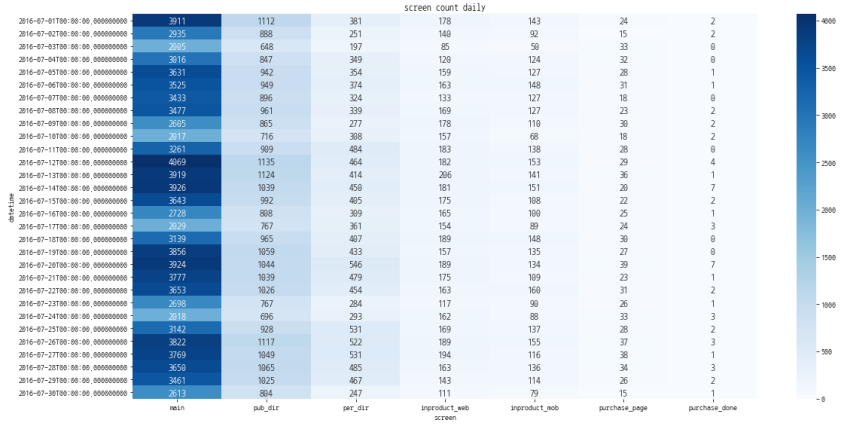
In [74]:

```
# heat map
screens = df.groupby(["datetime", "screen"])['sessionid'].nunique().unstack().fillna(0).astype(
3
4 # cols order change
5 screens = screens[screens.mean().sort_values(ascending=False).index]
6
7 screens[:10]
```

. . .

In [75]:

```
plt.subplots(figsize=(17,8))

sns.heatmap(screens, annot=True, fmt="d", annot_kws={"size": 12}, cmap='Blues');

plt.title("screen count daily")
plt.tight_layout()
```



Note.

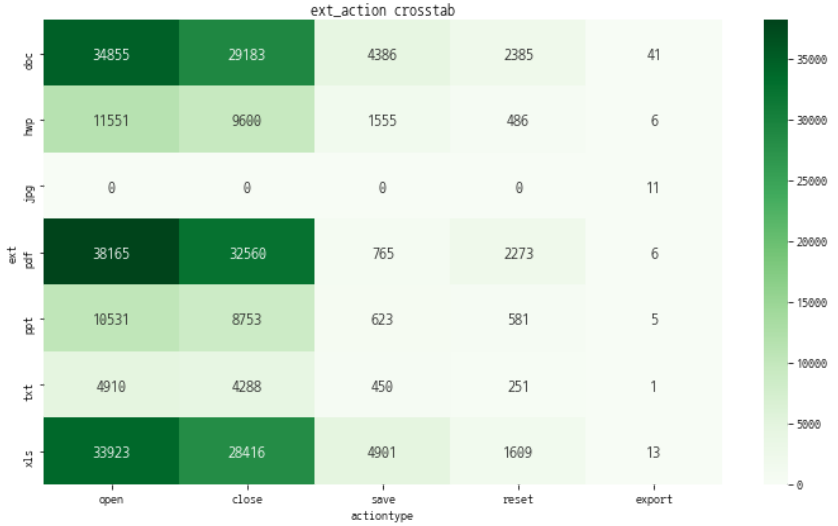
- doc, pdf, xls 순으로 주로 사용
- 주요 문서 이용 위치는 otherapp
- Main -> 구매완료(purchase_done) 까지 과정에서 대부분 이탈

3.2 Pivoting 을 통한 변수별 특성 탐색

- unstack, stack, pivot_table 과 같은 pandas 함수를 이용해, 다양한 각도에서 데이터 탐색

In [77]:

```
plt.subplots(figsize=(10,6))

ext_action = df.groupby(["ext", "actiontype"])['sessionid'].nunique().unstack().fillna(0).astyp
ext_action = ext_action[['open', 'close', 'save', 'reset', 'export']] # 행동 흐름 순대로 재정렬
sns.heatmap(ext_action, annot=True, fmt="d", annot_kws={"size": 12}, cmap='Greens');

plt.title("ext_action crosstab")
plt.tight_layout()
```



In [78]:

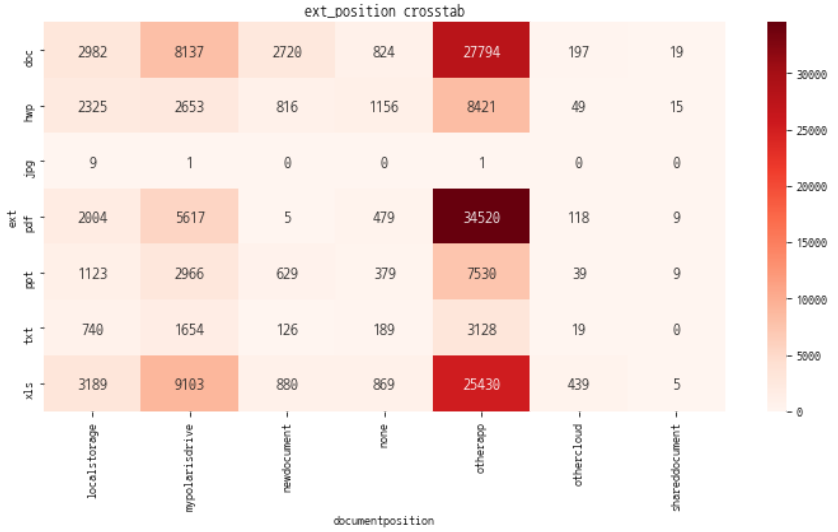
```
# count to percent
df.groupby(["ext", "actiontype"])['sessionid'].nunique().unstack().fillna(0).apply(lambda x: x/
```

Out[78]:

actiontype	close	export	open	reset	save
ext					
doc	25.87	49.40	26.02	31.44	34.59
hwp	8.51	7.23	8.62	6.41	12.26
jpg	0.00	13.25	0.00	0.00	0.00
pdf	28.87	7.23	28.50	29.97	6.03
ppt	7.76	6.02	7.86	7.66	4.91
txt	3.80	1.20	3.67	3.31	3.55
xls	25.19	15.66	25.33	21.21	38.65

In [36]:

```
plt.subplots(figsize=(10,6))

ext_pos = df.groupby(["ext", "documentposition"])['sessionid'].nunique().unstack().fillna(0).as

sns.heatmap(ext_pos, annot=True, fmt="d", annot_kws={"size": 12}, cmap='Reds');

plt.title("ext_position crosstab")
plt.tight_layout()
```



3.4 구간별 전환율 (Funnel) Daily Trend

In [89]:

```
screens.head(10) # daily session count by screen
```

...

In [93]:

```
# mean of each columns
conver_cnt = screens.mean().apply(lambda x: int(x)).sort_values(ascending=False)

conver_cnt
```

Out[93]:

```
screen
main             3255
pub_dir           939
per_dir           390
inproduct_web     161
inproduct_mob     119
purchase_page      27
purchase_done       1
dtype: int64
```

In [94]:

```
# average conversion rate
for i in range(len(conver_cnt)-1):
    print((conver_cnt[i+1] / (conver_cnt[i] * 1.0) * 100).round(2))
```

```
28.85
41.53
41.28
73.91
22.69
3.7
```

In [96]:

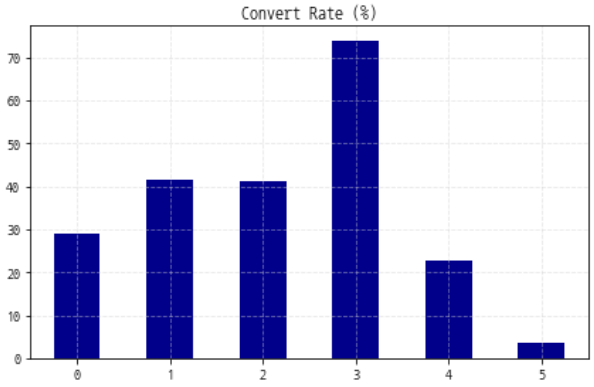
```
# another option for gettig average conversion rate
conver_rt = [(conver_cnt[i+1] / (conver_cnt[i] * 1.0) * 100).round(2) for i in range(len(conver

conver_rt
```

Out[96]:

```
[28.85, 41.53, 41.28, 73.91, 22.69, 3.7]
```

In [97]:

```
pd.Series(conver_rt).plot(kind='bar', color = 'darkblue', rot=0)

plt.title("Convert Rate (%)")
plt.grid(color='lightgrey', alpha=0.5, linestyle='--')
plt.tight_layout()
```



In [98]:

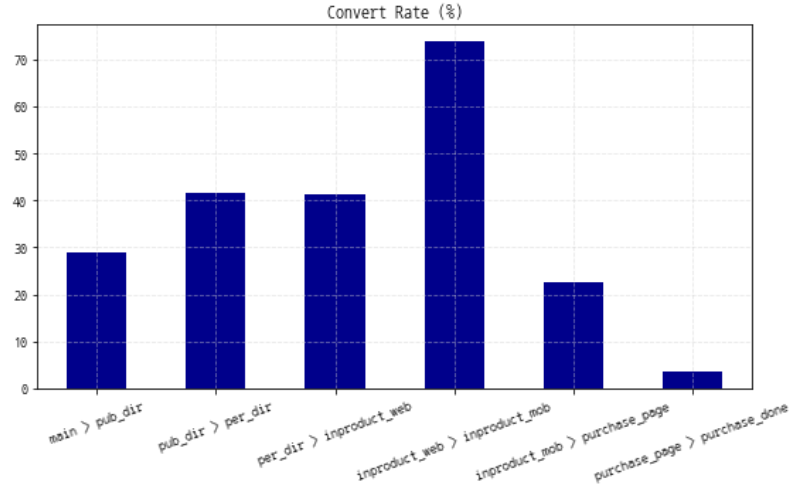
```
# create x labels
fun_label = [conver_cnt.index[k] + " > " + conver_cnt.index[k + 1] for k, v in enumerate(conver

fun_label
```

Out[98]:

```
['main > pub_dir',
 'pub_dir > per_dir',
 'per_dir > inproduct_web',
 'inproduct_web > inproduct_mob',
 'inproduct_mob > purchase_page',
 'purchase_page > purchase_done']
```

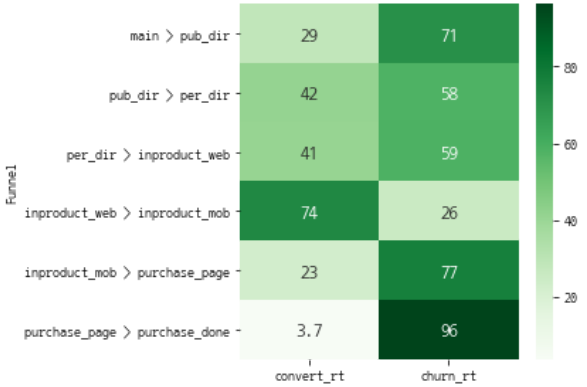
In [99]:

```
# with revised x labels
pd.Series(conver_rt, index=fun_label).plot(kind='bar', color = 'darkblue', rot=20, figsize=(8,5

plt.title("Convert Rate (%)")
plt.grid(color='lightgrey', alpha=0.5, linestyle='--')
plt.tight_layout()
```



In [100]:

```
# another visualization
conv_rt_tb = pd.Series(conver_rt, index=fun_label).to_frame()
conv_rt_tb.index.name = 'Funnel'
conv_rt_tb.columns = ['convert_rt']
conv_rt_tb['churn_rt'] = 100 - conv_rt_tb['convert_rt']

sns.heatmap(conv_rt_tb, annot=True, annot_kws={"size": 13}, cmap='Greens');

plt.tight_layout()
```



3.5 중간 정리

- **일별 주요 통계**
  - 활성화 세션의 경우 주말에 감소하고 주중에 증가하는 트렌드 보임
  - 확장자별 1 tier에는 pdf, xls, doc가 포지셔닝되며, 2 tier에는 hwp, ppt가 포함됨
  - 문서의 이용 위치는 'other app' 이 압도적으로 높음
  - 스크린별로 사용성 파악 결과, 메인(main) 화면이 가장 많이 노출되며 다음 화면(pub_dir or per_dir)으로 넘어가는 경우 많지 않음
- **구간별 전환율**
  - 전환율이 가장 낮은 구간(=이탈이 가장 높은 구간)은 구매정보 페이지에서 구매 완료 페이지로 전환하는 구간임(3.7%)
  - 제품내 웹 -> 앱으로 전환하는 구간은 전환율이 양호함(74%)

4. 클러스터링 For Targeting

In [101]:

```
df.head()
```

Out[101]:

	actiontype	ismydoc	ext	sessionid	documentposition	datetime	screen
0	open	noview	pdf	sess0	localstorage	2016-07-18	per_dir
1	close	noview	pdf	sess0	localstorage	2016-07-18	per_dir
2	open	view	pdf	sess0	mypolarisdrive	2016-07-18	pub_dir
3	close	view	pdf	sess0	mypolarisdrive	2016-07-18	pub_dir
4	open	noview	pdf	sess1	otherapp	2016-07-06	main

In [102]:

```
# 확장자만 기준으로 group by sessionid 하여 클러스터링을 위한 전처리 진행
# Note. if 다른 변수가 같이 있다면 scaling 필수
df_ext = df.query("actiontype == 'open'").groupby(["sessionid", "ext"]).size().unstack().fillna

df_ext.head(10)
```

...

In [103]:

```
df_ext_elbow = df_ext.copy()
```

클러스터별 전환율

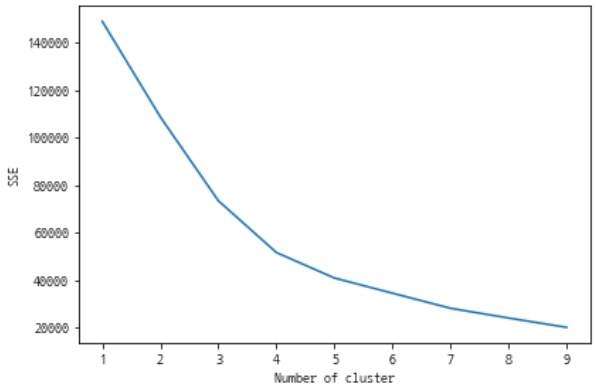
- 전체 유저를 대상으로 전환율을 구할 경우 큰 Implication 을 얻기 어렵다. 최대한 쪼개서 볼수록 많은 인사이트를 얻을 수 있으며, 유저를 분류할 때 많이 쓰이는 기법이 클러스터링이다.
- 클러스터링을 위한 주요 변수 설정 (필요시 PCA로 차원 축소)
- K-Means 알고리즘 적용후 그룹핑
- 그룹별 대푯값 및 시각화 등을 통해 탐색
- 여러 요인을 카테고리 단으로 묶는 요인분석(Factor Analysis)과 목적이 다름에 유의

In [104]:

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

In [105]:

```
# scree plot with sum of square error
sse = {}

for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(df_ext_elbow)
    df_ext_elbow["clusters"] = kmeans.labels_
    #print(data["clusters"])
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of samples to their closest cluster ce

plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.ylabel("SSE")
plt.tight_layout()
plt.show()
```



In [106]:

```
km = KMeans(n_clusters=4).fit(df_ext)
```

In [107]:

```
labels = km.labels_

labels
```

Out[107]:

```
array([2, 2, 0, ..., 3, 1, 2], dtype=int32)
```

In [108]:

```
df_ext['group'] = labels
```

In [109]:

```
df_ext.head()
```

...

In [110]:

```
df_ext.group.value_counts()
```

Out[110]:

```
2    35731
1    33269
3    30119
0    21538
Name: group, dtype: int64
```

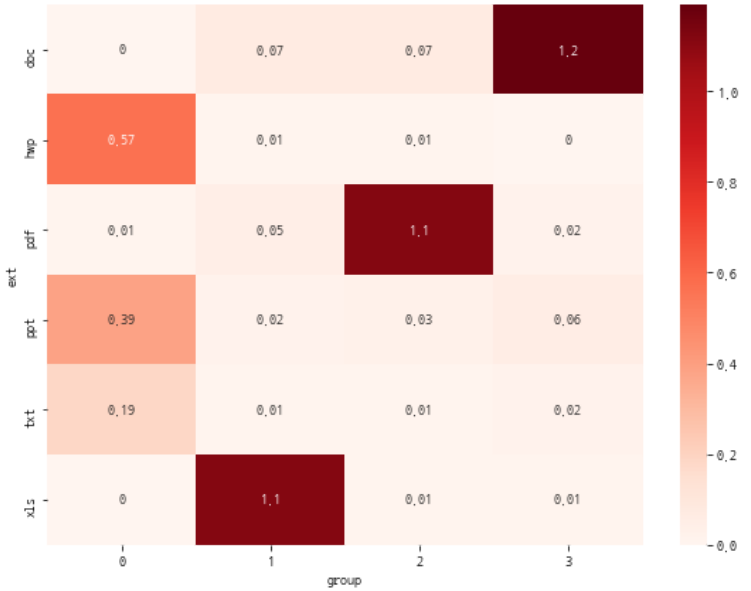
In [111]:

```
df_ext_mean = df_ext.groupby("group").mean().round(2)

df_ext_mean
```

...

In [112]:

```
plt.subplots(figsize=(8,6))

sns.heatmap(df_ext_mean.T, annot=True, cmap='Reds')

plt.tight_layout()
```



Note.

- Labeling 주의 (실행시마다 바뀜)

In [113]:

```
# be careful with the values; the group no. changes randomly
group_name = {0: 'gr_hwp',
              1: 'gr_xls',
              2: 'gr_pdf',
              3: 'gr_doc'}
```

In [114]:

```
df_ext['group'] = df_ext['group'].replace(group_name)
```

In [115]:

```
df_ext.head()
```

...

In [116]:

```
groups = df_ext.groupby("group")

fig, ax = plt.subplots()

for name, group in groups:
    ax.plot(group['doc'], group['hwp'], marker='o', linestyle='', ms=8, label=name)
    ax.legend()

plt.xlabel("doc")
plt.ylabel("hwp")
plt.grid(color='lightgrey', alpha=0.5, linestyle='--')
plt.show()
```

...

In [117]:

```
fig, ax = plt.subplots()

for name, group in groups:
    ax.plot(group['pdf'], group['xls'], marker='o', linestyle='', ms=8, label=name)
    ax.legend()

plt.xlabel("pdf")
plt.ylabel("xls")
plt.grid(color='lightgrey', alpha=0.5, linestyle='--')
plt.show()
```

...

In [118]:

```
df_open = df.query("actiontype == 'open'")

df_open.head(10)
```

...

In [119]:

```
df_ext.group.head()
```

Out[119]:

```
sessionid
sess0       gr_pdf
sess1       gr_pdf
sess10      gr_hwp
sess100     gr_xls
sess1000    gr_pdf
Name: group, dtype: object
```

In [120]:

```
df_cluster = df_open.merge(df_ext[['group']].reset_index(), on='sessionid', how='left')

df_cluster.head(10)
```

...

In [121]:

```
#define a function to get conversion rates
def conv_rt_by_grp(gr):
    df_gr_screen = df_cluster[df_cluster['group'] == gr]\ # gr명만으로 데이터 인덱싱
                     .groupby(["datetime", "screen"])['sessionid']\ # 날짜와, 스크린 기준으로 
                     .nunique().unstack().fillna(0).astype(int) # 재구조화

    conver_cnt = df_gr_screen.mean().apply(lambda x: int(x)).sort_values(ascending=False) # 스
    conver_rt = [conver_cnt[i+1] / (conver_cnt[i] * 1.0) * 100 for i in range(len(conver_cnt))
    # 앞화면 > 뒷화면으로 라벨 만들기
    fun_label = [conver_cnt.index[k] + " > " + conver_cnt.index[k + 1] for k, v in enumerate(co
    conver_rt = pd.Series(conver_rt, index=fun_label).fillna(0)

    return conver_rt
```

In [122]:

```
conv_rt_pdf = conv_rt_by_grp('gr_pdf')

conv_rt_pdf
```

Out[122]:

```
main > pub_dir                  15.80
pub_dir > per_dir               29.27
per_dir > inproduct_web         10.42
inproduct_web > purchase_page   40.00
purchase_page > purchase_done    0.00
dtype: float64
```

In [123]:

```
conv_rt_doc = conv_rt_by_grp('gr_doc')
conv_rt_doc
```

Out[123]:

```
main > pub_dir                 24.71
pub_dir > inproduct_web        35.26
inproduct_web > per_dir        88.06
per_dir > purchase_page         3.39
purchase_page > purchase_done   0.00
dtype: float64
```

In [124]:

```
conv_rt_xls = conv_rt_by_grp('gr_xls')
conv_rt_xls
```

Out[124]:

```
main > pub_dir                 32.83
pub_dir > per_dir              27.10
per_dir > inproduct_web        35.21
inproduct_web > purchase_page  48.00
purchase_page > purchase_done   0.00
dtype: float64
```

In [125]:

```
conv_rt_hwp = conv_rt_by_grp('gr_hwp')
conv_rt_hwp
```

Out[125]:

```
main > pub_dir                 32.02
pub_dir > per_dir              55.48
per_dir > inproduct_web        37.21
inproduct_web > purchase_page   6.25
purchase_page > purchase_done   0.00
dtype: float64
```

In [126]:

```
fig, ax = plt.subplots(4, 1, figsize=(8,20), sharey=True)

conv_rt_pdf.plot(kind='bar', ax=ax[0], color = 'brown', rot=30)
ax[0].set_title('PDF Group')
ax[0].set_ylabel('PDF Group')
ax[0].grid(color='lightgrey', alpha=0.5, linestyle='--')

conv_rt_doc.plot(kind='bar', ax=ax[1], color = 'green', rot=30)
ax[1].set_ylabel('DOC Group')
ax[1].set_title('DOC Group')
ax[1].grid(color='lightgrey', alpha=0.5, linestyle='--')

conv_rt_xls.plot(kind='bar', ax=ax[2], color = 'purple', rot=30)
ax[2].set_ylabel('XLS Group')
ax[2].set_title('XLS Group')
ax[2].grid(color='lightgrey', alpha=0.5, linestyle='--')

conv_rt_hwp.plot(kind='bar', ax=ax[3], color = 'darkblue', rot=30)
ax[3].set_ylabel('HWP Group')
ax[3].set_title('HWP Group')
ax[3].grid(color='lightgrey', alpha=0.5, linestyle='--')

plt.tight_layout()
```

...

In [127]:

```
# 그룹별 평균 전환율
gr_pdf_avg = conv_rt_pdf.replace(0, np.nan).mean()
gr_doc_avg = conv_rt_doc.replace(0, np.nan).mean()
gr_xls_avg = conv_rt_xls.replace(0, np.nan).mean()
gr_hwp_avg = conv_rt_hwp.replace(0, np.nan).mean()

print("pdf", gr_pdf_avg)
print("doc", gr_doc_avg)
print("xls", gr_xls_avg)
print("hwp", gr_hwp_avg)
```

```
pdf 23.871143498284695
doc 37.85502552985396
xls 35.78564611183911
hwp 32.741991670438274
```

In [129]:

```
# 가중치 임의 설정
weights = [1, 1.3, 1.5, 2, 2.5]

# 가중 평균
def weight_avg(gr):
    w_avg = (gr.values * weights).sum() / len(gr)
    return w_avg
```

In [130]:

```
gr_pdf_w = weight_avg(conv_rt_pdf)
gr_doc_w = weight_avg(conv_rt_doc)
gr_xls_w = weight_avg(conv_rt_xls)
gr_hwp_w = weight_avg(conv_rt_hwp)

print("pdf:", gr_pdf_w)
print("doc:", gr_doc_w)
print("xls:", gr_xls_w)
print("hwp:", gr_hwp_w)
```

```
pdf: 29.894679026270033
doc: 41.883746148516025
xls: 43.37559784850795
hwp: 34.49355582697327
```

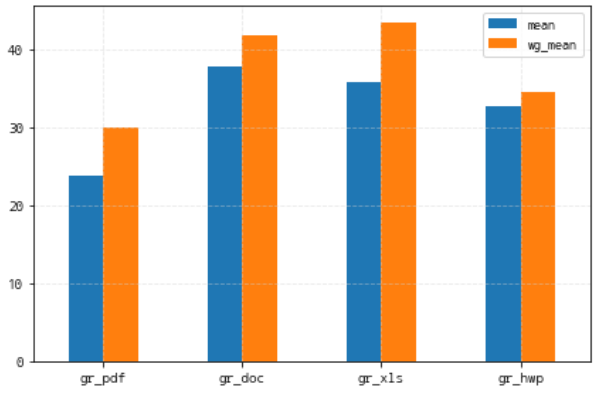
In [131]:

```
avg_df = pd.DataFrame(list(zip([gr_pdf_avg, gr_doc_avg, gr_xls_avg, gr_hwp_avg],\
                                [gr_pdf_w, gr_doc_w, gr_xls_w, gr_hwp_w])), \
                      columns = ['mean', 'wg_mean'],\
                      index = ['gr_pdf', 'gr_doc', 'gr_xls', 'gr_hwp'])

avg_df
```

...

In [132]:

```
avg_df.plot(kind='bar', fontsize=11, rot=0)

plt.grid(color='lightgrey', alpha=0.5, linestyle='--')
plt.tight_layout()
```



- 현황
  - 주말 대비 주중의 사용성 높음 (세션수 기준)
  - doc, pdf, xls 확장자가 주요 항목
  - 전체적으로 구매정보 -> 구매결제로 이어지는 전환율이 낮아(3.7%) 개선이 시급함
  - 바로 전단계인 제품내 -> 구매정보로의 전환율 역시 낮은 편임(23%)
- 시사점
  - 클러스터링 결과 PDF 사용 그룹의 전환율 상대적으로 낮아 원인 파악 필요(24%, 타그룹 평균 약35%)
  - 모든 funnel 단계에서 전환율 전반적으로 낮음 (약 20%대)
  - 개선 우선순위가 가장 높은(핵심)그룹으로 고려 가능
  - PDF 관련 기능 검토 필요 (안정성 등)
- 개선 방안
  - 제품 내에서 Right (timing, persons, contents) 제공하여 구매 정보 제고 및 구매 전환 유도
  - 디자인/컨텐츠 시안, 타깃 적절히 설계하여 a/b test 진행 후 개선 프로세스 반복
  - 관련 신규 부가 기능 검토 및 안정성 확보 필요
- 분석 한계점
  - 클러스터링 방식의 고도화 (보다 세분화된 그룹핑 및 전략 개발)

In [ ]:

```
1
```