





분석보고서

1. 프로젝트 개요

1-1. 주제



2호점 오픈 시, '커피템플'의 브랜드 가치와 매출 향상을 위한 솔루션 제안
(오프라인 매출 관련 데이터 기반 근거 도출)

 김사홍 국가대표 바리스타와 함께하는 Coffee Class 

1-2. 주제 선정의 배경

1. 오프라인 매장의 한계

- 현 오프라인 매장에서 나올 수 있는 매출은 한계가 있음
- 손님이 많은 경우 현재도 자리가 없어서 커피를 마시려면 웨이팅이 있음
- 현재 부지의 분계서 해당 부지 개발에 회의적이기 때문에 매장 확장의 여지가 매우 적음

2. 김사홍 대표님의 영향력

- 제주도 카페 중 김사홍 대표님을 아시거나 영향을 받은 분들이 많음
- 게스트 바리스타 현재 40회 정도 진행(정기 25차)

3. 김사홍 대표님의 가치관(철학)

a) 커피(원두)에 대한 철학

- 커피가 일상에서 중요한 것이라면 오프라인 경험을 통해 커피 그 이상의 가치(진정성)를 보여줘야 함
- 고객에게 좋은 원두와 커피를 제공하여 진정성을 전달하고 즐거움과 활력을 제공하여 유입을 늘리는 것

b) 커피템플 카페의 비전

- 커피템플을 제주 안에서 커피/카페하면 바로 떠오르는 'One and Only'로 자리매김하길 원함
- 우리나라 사람들 뿐만 아니라 외국인도 찾아오게 만들어 시장을 키우고 싶음(아직은 많지 않지만 한국인 친구 소개를 통한 독일인, 프랑스인 친구의 방문, 중국 관광객들의 유입 등)

4. 커피클래스 타당성

- 게스트 바리스타 in 커피템플
 - 현재도 게스트 바리스타를 통해 커피에 대한 새로운 경험을 제공하고 팬덤을 위한 이벤트를 꾸준히 진행하고 있음
 - 다양한 지역을 돌아다니며 게스트 바리스타 활동을 통해 팬덤 확보를 위해 좋은 시도이지만 제주를 떠올렸을 때 랜드마크 카페가 되기 위해서는 인하우스의 규칙적인 이벤트 진행이 반드시 필요함
- 오프라인 매장 집중전략
 - 대표님의 바리스팅 실력과 다양한 레시피
 - '커피의 가치와 진정성'에 대한 직원들과의 가치관 공유
 - 가치 공유를 통한 높은 수준의 고객 응대 서비스
- 매장 확장 (구두합의 완료)

- 내년 3월 확장 예정
- 위치는 공항 앞 GS칼텍스 부지로 기존 매장에 비해 접근성이 높아짐
- 1층은 드라이브 스루, 2층은 75평 평수의 큰 규모의 카페로 구성 예정

2. 프로젝트 팀 구성 및 역할

팀장 : 이양석

팀원 : 김병우, 송유림, 황상엽, 황소연

역할

- 내부 데이터 탐색(EDA) : 김병우, 송유림, 이양석, 황상엽, 황소연
- 주제선정을 위한 기획안 작성 : 김병우, 송유림, 이양석, 황상엽, 황소연
- 외부 자료(데이터) 조사 : 김병우, 송유림, 이양석, 황상엽, 황소연
- 변수 탐색/분석
 1. 날씨(기온, 강수량, 풍속) - 김병우, 송유림
 2. 주말/평일(요일) - 황소연
 3. 월별 입도객(관광객) - 황상엽
 4. 연휴 이벤트 - 이양석
- 모델링 : 김병우, 송유림, 황상엽
- 크롤링 기반 감성분석 : 김병우, 송유림, 황상엽
- 솔루션 제안 : 김병우, 송유림, 이양석, 황상엽, 황소연
- 데이터 기반 근거 탐색 : 김병우, 송유림, 이양석, 황상엽, 황소연
- 분석보고서 작성 : 김병우, 이양석, 황상엽
- 발표자료 작성 : 송유림, 황소연
- 프로젝트 발표 : 송유림

3. 프로젝트 수행 절차 및 방법

프로젝트 수행절차



3-1. 데이터 설명 (데이터 출처, 데이터 개요)

크롤링

활용 데이터 경로 주소 커피템플 네이버 지도 리뷰:

<https://pcmap.place.naver.com/restaurant/1275841339/review/visitor?from=map&fromPanelNum=2&reviewSort=recent&ts=1690651677592>

코드

네이버 지도 리뷰 크롤링 주피터.ipynb

▼ 네이버 지도 리뷰 크롤링 [주피터 환경]

```
!pip install beautifulsoup4
!pip install requests

from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
import pandas as pd

driver = webdriver.Chrome()
driver.get('https://pcmap.place.naver.com/restaurant/1275841339/review/visitor?from=map&fromPanelNum=2&reviewSort=recent&ts=1690651677592')

time.sleep(3) # 웹사이트가 완전히 로딩될 때까지 기다림

wait = WebDriverWait(driver, 10)

while True:
    try:
        # 리뷰 내의 '더보기' 버튼을 모두 클릭
        review_more_buttons = wait.until(EC.presence_of_all_elements_located((By.CSS_SELECTOR, 'div.ZZ40K > a > span.rvCSr > svg')))
        for more_button in review_more_buttons:
            more_button.click()
            time.sleep(1)
    except:
        pass # '더보기' 버튼이 없으면 다음 단계로 이동

    try:
```

```

# 페이지 하단의 '더보기' 버튼을 클릭하면서 추가 리뷰를 로드
page_more_button = wait.until(EC.presence_of_element_located((By.CSS_SELECTOR, 'div.lfh30 > a > svg')))
page_more_button.click()
time.sleep(2) # 로딩 시간 조정
except:
    break # '더보기' 버튼이 더 이상 없으면 종료

html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')

# CSS selector를 이용하여 원하는 데이터 위치 찾기
reviews = soup.select('#app-root > div > div > div:nth-child(7) > div:nth-child(3) > div.place_section.k5tcc > div.place_sec
visits = soup.select('div._7kR3e > span:nth-child(2)')
visit_times = soup.select('div._7kR3e > span:nth-child(1) > time') # 방문 일시
verification_methods = soup.select('div._7kR3e > span:nth-child(3)') # 인증 수단

# 각 리뷰, 방문 횟수, 방문 일시, 인증 수단을 저장하기 위한 리스트 생성
review_list = []
visit_list = []
visit_time_list = []
verification_list = []

# 각 데이터를 해당 리스트에 추가
for review, visit, visit_time, verification in zip(reviews, visits, visit_times, verification_methods):
    review_list.append(review.text)
    visit_list.append(visit.text)
    visit_time_list.append(visit_time.text)
    verification_list.append(verification.text)

# 데이터 프레임을 생성하고 CSV 파일로 저장
df = pd.DataFrame({
    'Review': review_list,
    'Visit Count': visit_list,
    'Visit Time': visit_time_list,
    'Verification Method': verification_list
})

df.to_csv('reviews.csv', index=False, encoding='utf-8-sig')

driver.quit() # 모든 작업이 끝나면 webdriver를 종료

```

사용 데이터

1. 커피템플 상세 데이터

→ 공휴일, 평일, 주말 평균 매출, 월별 메뉴 TOP5 시각화, 시간별 매출 합계 시각화 등 활용

2. 날씨(기온) 데이터

→ 커피템플 상세 데이터와 함께 일 매출과 기온 그래프 활용

3. 업종별 관광객 소비 데이터

→ 업종별 소비 금액 파악

4. 월별 제주 관광객입도 데이터

→ 평균 요일별 입도객 시각화, 날짜별 매출과 관광객 수 시각화

5. 출입국 관광 통계 데이터

→ 날짜별 매출과 관광객 수 시각화

데이터 출처

→ 5-2 항목에 기입

3-2. 데이터 샘플

- 커피템플 상세 데이터

커피템플 상세 데이터 전처리완료.xlsx

```
df.head(), df.info()
```

	결제일	결제시간	카테고리	상품명	옵션	수량	상품별	단가	상품별	합계
0	2022-02-28	17:41:31	비버리지	따뜻한 어린이 우유	-	2	0	0	0	0
1	2022-02-28	16:47:59	디저트	포장	-	1	0	0	0	0
2	2022-02-28	16:27:37	핸드드립	야외	-	1	0	0	0	0
3	2022-02-28	16:25:58	비버리지	야외	-	1	0	0	0	0
4	2022-02-28	16:25:01	비버리지	야외	-	1	0	0	0	0

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 143016 entries, 0 to 143015
Data columns (total 8 columns):
#   Column      Non-Null Count  Dtype
---  -
0   결제일      143016 non-null object
1   결제시간    143016 non-null object
2   카테고리    143016 non-null object
3   상품명      143016 non-null object
4   옵션        143016 non-null object
5   수량        143016 non-null int64
6   상품별 단가  143016 non-null int64
7   상품별 합계  143016 non-null int64
dtypes: int64(3), object(5)
memory usage: 8.7+ MB
```

- 월별 제주 입도객

```
df.head(), df.info()
```

	날짜	외국인	내국인	총계	전년	증감률(%)
0	2022-02-01	3148	1026355	1029503	793768	29.698224
1	2022-03-01	3258	869828	873086	893326	-2.265690
2	2022-04-01	3687	1174769	1178456	1082861	8.828003
3	2022-05-01	4574	1301963	1306537	1136452	14.966316
4	2022-06-01	5622	1277848	1283470	1138867	12.697093

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16 entries, 0 to 15
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   날짜        16 non-null    datetime64[ns]
1   외국인      16 non-null    int64
2   내국인      16 non-null    int64
3   총계        16 non-null    int64
4   전년        16 non-null    int64
5   증감률(%)   16 non-null    float64
dtypes: datetime64[ns](1), float64(1), int64(4)
memory usage: 896.0 bytes
```

- 날씨 + 커피템플

[DAS]FP_제주시_날씨+커피템플.xlsx

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 453 entries, 0 to 457
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   일시        453 non-null    object
1   일매출      453 non-null    float64
2   기온(°C)    453 non-null    float64
3   강수량(mm)  453 non-null    float64
dtypes: float64(3), object(1)
memory usage: 17.7+ KB
```

- 휴일, 공휴일, 평일 평균 매출

```
average_sales_by_type = average_sales_by_type.reset_index()
average_sales_by_type.columns = ['일자 유형', '일별 평균 매출']

average_sales_by_type, average_sales_by_type.shape
```

- 월별 메뉴

월별메뉴_전처리완.csv

```
df = pd.read_csv('/content/월별메뉴_전처리완.csv')
df.head(), df.shape
```

- 일별 제주 입도객

[DAS]FP_일별제주입도객_0804.xlsx

```
(
   일자  일별 입도객  월  요일
0  2022-06-01    46660   6    2
1  2022-06-02    42802   6    3
2  2022-06-03    45512   6    4
3  2022-06-04    49374   6    5
4  2022-06-05    40101   6    6,
(182, 4))
```

- 월별 해외관광객, 제주관광객

	날짜	해외관광객	연	월	관광객
0	2022-02-01	112722	2022	2	1029503
1	2022-03-01	145503	2022	3	873086
2	2022-04-01	215246	2022	4	1178456
3	2022-05-01	315945	2022	5	1306537
4	2022-06-01	412798	2022	6	1283470

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16 entries, 0 to 15
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   날짜        16 non-null     datetime64[ns]
1   해외관광객  16 non-null     int64
2   연          16 non-null     int64
3   월          16 non-null     int64
4   관광객      16 non-null     int64
dtypes: datetime64[ns](1), int64(4)
memory usage: 768.0 bytes
```

- 매출과 업종별 관광객 소비금액

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16 entries, 0 to 15
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   연          16 non-null     int64
1   월          16 non-null     int64
2   매출        16 non-null     int64
3   기준연월    16 non-null     datetime64[ns]
4   전체업종소비금액  16 non-null     int64
5   식음료업소비금액  16 non-null     int64
dtypes: datetime64[ns](1), int64(5)
memory usage: 896.0 bytes
```

3-3. 데이터 수집 및 전처리

상세_데이터_전처리.ipynb

▼ 상세 데이터 전처리

```
# 세 개의 엑셀 파일을 각각 데이터프레임으로 읽어옵니다.
file1 = '커피템플_상세_데이터_220201_220228.xlsx'
file2 = '커피템플_상세_데이터_220301_220331.xlsx'
file3 = '커피템플_상세_데이터_220401_220430.xlsx'
file4 = '커피템플_상세_데이터_220501_220531.xlsx'
file5 = '커피템플_상세_데이터_220601_220630.xlsx'
file6 = '커피템플_상세_데이터_220701_220731.xlsx'
file7 = '커피템플_상세_데이터_220801_220831.xlsx'
file8 = '커피템플_상세_데이터_220901_220930.xlsx'
file9 = '커피템플_상세_데이터_221001_221031.xlsx'
file10 = '커피템플_상세_데이터_221101_221130.xlsx'
file11 = '커피템플_상세_데이터_221201_221231.xlsx'
file12 = '커피템플_상세_데이터_230101_230131.xlsx'
file13 = '커피템플_상세_데이터_230201_230228.xlsx'
file14 = '커피템플_상세_데이터_230301_230331.xlsx'
file15 = '커피템플_상세_데이터_230401_230430.xlsx'
file16 = '커피템플_상세_데이터_230501_230531.xlsx'
```

```

df1 = pd.read_excel(file1)
df2 = pd.read_excel(file2)
df3 = pd.read_excel(file3)
df4 = pd.read_excel(file4)
df5 = pd.read_excel(file5)
df6 = pd.read_excel(file6)
df7 = pd.read_excel(file7)
df8 = pd.read_excel(file8)
df9 = pd.read_excel(file9)
df10 = pd.read_excel(file10)
df11 = pd.read_excel(file11)
df12 = pd.read_excel(file12)
df13 = pd.read_excel(file13)
df14 = pd.read_excel(file14)
df15 = pd.read_excel(file15)
df16 = pd.read_excel(file16)

# 데이터프레임을 리스트로 저장합니다.
dataframes = [df1, df2, df3, df4, df5, df6, df7, df8, df9, df10, df11, df12, df13, df14, df15, df16]

# pd.concat() 함수를 사용하여 데이터를 결합합니다.
combined_df = pd.concat(dataframes)

# 결합된 데이터프레임을 새로운 엑셀 파일로 저장합니다.
combined_df.to_excel('커피템플_상세_데이터_전체.xlsx', index=False)

# 데이터 전처리 함수를 적용하여 새 데이터셋을 만듭니다.
all_data_preprocessed = preprocess_data(all_data)

# 유지할 열을 정의합니다.
columns_to_keep = ['결제일', '결제시간', '카테고리', '상품명', '옵션', '수량', '상품별_단가', '상품별_합계']

# 데이터 전처리 단계를 각 데이터셋에 적용하는 함수를 정의합니다.
def preprocess_data(df):
    # 지정된 열만 유지합니다.
    df = df[columns_to_keep]

    # 누락된 '결제일'과 '결제시간'을 이전 행의 값으로 대체합니다.
    df['결제일'].fillna(method='ffill', inplace=True)
    df['결제시간'].fillna(method='ffill', inplace=True)

    # 환불 주문을 제거합니다. (상품별_합계가 '-'인 경우 환불 주문으로 판단합니다)
    df = df[df['상품별_합계'] != '-']

    return df

all_data_preprocessed = preprocess_data(all_data)

all_data_preprocessed.head()

```

[DAS]FP_날씨데이터(기온,강수)_전처리.ipynb

▼ 날씨데이터(기온, 강수량) 전처리

```

import pandas as pd
from google.colab import drive
drive.mount('/content/drive')

# 22.02.10 - 22.12.31 data 불러오기
w1 = pd.read_csv('/content/drive/SharedDrives/[DAS]FinalProject_5조/송유림/날씨데이터/제주시_기온+강수_22.02.10-22.12.31.csv', encoding='c

# 23.01.01 - 23.05.31 data 불러오기
w2 = pd.read_csv('/content/drive/SharedDrives/[DAS]FinalProject_5조/송유림/날씨데이터/제주시_기온+강수_23.01.01-23.05.31.csv', encoding='c

# w1, w2 concat
w = pd.concat([w1, w2])

# 지점, 지점명 컬럼 삭제
w = w.drop(['지점', '지점명'], axis=1)

# '일시' 컬럼에서 시간 삭제
w['일시'] = w['일시'].str[:10]

```



```
# 강수량이 0.0인 row 개수 확인 -> 505개 -> 0.0은 강수량이 0.0 ~ 0.1 사이의 값일 때 0.0으로 표기함
w[w['강수량(mm)'] == 0.0].shape[0]

# 0.0 -> 0.05로 변경(0.0 ~ 0.1의 중앙값으로 대체)
w.loc[w['강수량(mm)'] == 0.0, '강수량(mm)'] = 0.05

# 재확인
w[w['강수량(mm)'] == 0.0].shape[0]

# null값 개수 확인 -> 10,192개 -> 비가 아예 오지 않음을 뜻함
w[w['강수량(mm)'].isnull()].shape[0]

# null -> 0.0으로 변경
w.loc[w['강수량(mm)'].isnull(), '강수량(mm)'] = 0.0

# 재확인
w[w['강수량(mm)'].isnull()].shape[0]

# 일별 평균 기온
w1 = w.groupby('일시')['기온(°C)'].mean().round(1).reset_index()

# 일별 총 강수량
w2 = w.groupby('일시')['강수량(mm)'].sum().round(1).reset_index()

df = pd.merge(w1, w2, on='일시', how='outer')
```

[DAS]FP_날씨+커피템플_완성본.ipynb

▼ 기온 + 커피템플

```
w = pd.read_csv('/content/drive/Shareddrives/[DAS]FinalProject_5조/송유림/제주시_날씨(기온, 강수량)_완성본.csv')

coffee = pd.read_excel('/content/drive/Shareddrives/[DAS]FinalProject_5조/송유림/커피템플_상세_데이터_전처리완료.xlsx')

# 일자별 매출 합계
c = coffee.groupby('결제일')['상품별 합계'].sum().reset_index()

# rename -> merge하기 위해
c = c.rename(columns={'결제일': '일시', '상품별 합계': '일매출'})

# 날씨+커피템플
df = pd.merge(c, w, on='일시', how='outer')

# 일매출이 100,000원 미만인 데이터 -> 이상치로 간주 -> 제거
df = df[df['일매출'] >= 100000]

# '일시' 컬럼을 object -> datetime 형변환
df['일시'] = pd.to_datetime(df['일시'])

# 일매출 컬럼을 float -> int로 형변환
df['일매출'] = df['일매출'].astype(int)

# 그래프 그리기
fig, ax1 = plt.subplots(figsize = (20, 6))

# 첫 번째 축 (일매출)
ax1.plot(df['일시'], df['일매출'], label='일매출', color='tab:gray', linestyle='--')
ax1.set_xlabel('일시')
ax1.set_ylabel('일매출', color='tab:gray')
ax1.tick_params(axis='y', labelcolor='tab:gray')

# 두 번째 축 (기온)
ax2 = ax1.twinx()
ax2.plot(df['일시'], df['기온(°C)'], label='기온(°C)', color='tab:red', linestyle='--')
ax2.set_ylabel('기온(°C)', color='tab:red')
ax2.tick_params(axis='y', labelcolor='tab:red')

plt.title('일매출과 기온 그래프')
plt.grid(True)
plt.xticks(rotation=45)

# 범례 표시
```

```

lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax2.legend(lines + lines2, labels + labels2, loc='upper left')

plt.show()

```

▼ 강수량 시각화(사용 x)

```

# 그래프 그리기
fig, ax1 = plt.subplots(figsize = (20, 6))

# 첫 번째 축 (일매출)
ax1.plot(df['일시'], df['일매출'], label='일매출', color='tab:gray', linestyle='-.')
ax1.set_xlabel('일시')
ax1.set_ylabel('일매출', color='tab:gray')
ax1.tick_params(axis='y', labelcolor='tab:gray')

# 두 번째 축 (강수량)
ax2 = ax1.twinx()
ax2.plot(df['일시'], df['강수량(mm)'], label='강수량(mm)', color='tab:blue', linestyle='-')
ax2.set_ylabel('강수량(mm)', color='tab:blue')
ax2.tick_params(axis='y', labelcolor='tab:blue')

plt.title('일매출과 강수량 그래프')
plt.grid(True)
plt.xticks(rotation=45)

# 범례 표시
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax2.legend(lines + lines2, labels + labels2, loc='upper left')

plt.show()

```

요일별 평균 입도객.ipynb

▼ 요일별 평균 입도객

```

# 데이터 불러오기
df = pd.read_excel("/content/[DAS]FP_일별제주입도객_0804.xlsx")
df['월'] = df['일자'].dt.month
df['요일'] = df['일자'].dt.dayofweek

# x축 라벨
x_labels = ['월', '화', '수', '목', '금', '토', '일']

# 지정된 월에 대한 요일별 평균 입도객 그래프 생성 함수
def plot_avg_visitors_by_month(year, month, ax):
    # 해당 월 데이터 필터링
    monthly_data = df[(df['일자'].dt.year == year) & (df['일자'].dt.month == month)]

    # 요일별 평균 계산
    avg_by_day = monthly_data.groupby('요일')['일별 입도객'].mean()

    # 바 그래프 그리기
    avg_by_day.plot(kind='bar', color='skyblue', width=0.7, ax=ax)

    # 그래프 설정
    ax.set_title(f"{year}년 {month}월 요일별 입도객")
    ax.set_xlabel("")
    ax.set_ylabel("평균 입도객 수")
    ax.set_ylim(20000, avg_by_day.max() + 5000)
    ax.set_xticks(range(7))
    ax.set_xticklabels(x_labels, rotation=0)
    ax.grid(True, axis='y', linestyle='--', linewidth=0.7, alpha=0.7)

    # 막대 위에 평균 입도객 수 표시
    for p in ax.patches:
        ax.annotate(f"{int(p.get_height()):,}", (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='center', xytext=(0, 10), textcoords='offset points')

# 전체 그래프 생성

```

```
fig, axes = plt.subplots(3, 2, figsize=(15, 18))
for (year, month), ax in zip(specified_months, axes.ravel()):
    plot_avg_visitors_by_month(year, month, ax)
plt.tight_layout()
plt.show()
```

주말,공휴일,평일 평균매출.ipynb

▼ 주말, 공휴일, 평일 평균 매출

```
# 엑셀 파일 읽기
data = pd.read_excel("/content/커피애프롬_상세_데이터_전처리완료.xlsx")

# 공휴일 리스트 생성
holidays = [
    "2022-01-29", "2022-01-30", "2022-01-31", "2022-02-01", "2022-02-02",
    "2022-02-26", "2022-02-27", "2022-02-28", "2022-03-01",
    "2022-03-09",
    "2022-05-05", "2022-05-06", "2022-05-07", "2022-05-08",
    "2022-06-01",
    "2022-06-04", "2022-06-05", "2022-06-06",
    "2022-08-13", "2022-08-14", "2022-08-15",
    "2022-09-09", "2022-09-10", "2022-09-11", "2022-09-12",
    "2022-10-01", "2022-10-02", "2022-10-03",
    "2022-10-08", "2022-10-09", "2022-10-10",
    "2023-01-21", "2023-01-22", "2023-01-23", "2023-01-24",
    "2023-03-01",
    "2023-05-05", "2023-05-06", "2023-05-07",
    "2023-05-27", "2023-05-28", "2023-05-29"
]

# 날짜를 datetime 형식으로 변환
data['결제일'] = pd.to_datetime(data['결제일'])

# 공휴일, 평일, 주말 데이터 분류
data['일자 유형'] = data['결제일'].apply(lambda x: '공휴일' if x.strftime('%Y-%m-%d') in holidays else ('주말' if x.weekday() >= 5 else '평일'))

# 상품별 합계를 이용하여 일자 유형별 매출 합계 계산
sales_by_type = data.groupby('일자 유형')['상품별 합계'].sum()

# 일자 유형별 날짜 개수 계산
days_by_type = data['결제일'].groupby(data['일자 유형']).nunique()

# 일자 유형별 평균 매출 계산
average_sales_by_type = sales_by_type / days_by_type

# 막대 그래프 시각화
fig, ax = plt.subplots(figsize=(10, 6))
average_sales_by_type.plot(kind='bar', color=['red', 'blue', 'green'], ax=ax)

# 타이틀 및 라벨 설정
plt.title("일자 유형별 일별 평균 매출")
plt.xlabel("일자 유형")
plt.ylabel("일별 평균 매출")
plt.xticks(rotation=0)

# 각 막대에 평균 매출값 표시
for p in ax.patches:
    ax.annotate(f"{int(p.get_height()):,}원", (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 10), textcoords='offset points')

plt.tight_layout()

# 그래프 출력
plt.show()
```

[DAS]FP 월별메뉴 전처리_0805.ipynb

▼ 월별메뉴 전처리

```
df = pd.read_excel('/content/drive/SharedDrives/[DAS]FinalProject_5조/송유림/월별메뉴/오프라인데이터_전처리완.xlsx')

#전처리
df = df[['결제일', '상품명', '상품별 합계']]

# 합계 0원 제외
df = df[df['상품별 합계'] != 0]

#메뉴 이름 통합
df['상품명'].replace('주시_아이스 텀라', '(I)텐저린 라떼', inplace=True)
df['상품명'].replace('아이스 텀저린 라떼', '(I)텐저린 라떼', inplace=True)

df['상품명'].replace('주시_텀저린 카푸치노', '(H)텀저린 카푸치노', inplace=True)
df['상품명'].replace('텀저린카푸치노', '(H)텀저린 카푸치노', inplace=True)

df['상품명'].replace('아이스 유자 아메리카노', '(I)유자 아메리카노', inplace=True)
df['상품명'].replace('주시_유자 아메리카노', '(I)유자 아메리카노', inplace=True)

df['상품명'].replace('클래식_(I) 아메리카노', '(I)아메리카노 클래식', inplace=True)
df['상품명'].replace('(I) 아메리카노 _ 클래식', '(I)아메리카노 클래식', inplace=True)

df['상품명'].replace('클래식_아메리카노', '(H)아메리카노 클래식', inplace=True)
df['상품명'].replace('아메리카노 _ 클래식', '(H)아메리카노 클래식', inplace=True)

df['상품명'].replace('주시_(I) 아메리카노', '(I)아메리카노 주시', inplace=True)
df['상품명'].replace('(I) 아메리카노 _ 주시', '(I)아메리카노 주시', inplace=True)

df['상품명'].replace('주시_아메리카노', '(H)아메리카노 주시', inplace=True)
df['상품명'].replace('아메리카노 _ 주시', '(H)아메리카노 주시', inplace=True)

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

리뷰 전처리.ipynb

▼ 리뷰 전처리(사용x)

```
import pandas as pd

df = pd.read_csv('/content/네이버_지도_리뷰(2020.12.06부터).csv')

# 전처리
df['Visit Count'] = df['Visit Count'].str.replace('번째 방문', '').astype(int)

df['Visit Time'] = df['Visit Time'].apply(lambda x: '23.' + x if x[:2] != '22' else x)

df['Verification Method'] = df['Verification Method'].str.replace('인증 수단', '')

# 이름 변경
df = df.rename(columns={'Review': '리뷰', 'Visit Count': '방문 횟수', 'Visit Time': '리뷰 작성 일자', 'Verification Method': '인증 수단'})

df.to_csv('/content/전처리_네이버_지도_리뷰.csv', index=False, encoding='utf-8-sig')
```

감성분석.ipynb

▼ 감성분석(사용x)

```

# pandas 라이브러리를 임포트합니다.
import pandas as pd

# 파일 경로
file_path = "C:/Users/user/크롤링/(라벨)커피템플_네이버리뷰_전처리완.xlsx"
# 파일을 불러옵니다.
data = pd.read_excel(file_path)

# 데이터의 처음 5행을 출력합니다.
data.head()

# null 값 제거
data = data.dropna()

import pandas as pd
from konlpy.tag import Okt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# 데이터 로딩
data = pd.read_excel('(라벨)커피템플_네이버리뷰_전처리완.xlsx')
data = data.dropna() # null 값 제거

reviews = data['리뷰'].tolist()
labels = data['label'].tolist()

# 텍스트 토큰화
okt = Okt()
reviews_tokenized = [' '.join(okt.morphs(text)) for text in reviews]

# 텍스트 벡터화
vectorizer = TfidfVectorizer()
reviews_vec = vectorizer.fit_transform(reviews_tokenized)

# 데이터셋 분리
x_train, x_test, y_train, y_test = train_test_split(reviews_vec, labels, test_size=0.2, random_state=42)

# 모델 학습
model = LogisticRegression()
model.fit(x_train, y_train)

# 모델 평가
preds = model.predict(x_test)
print("Accuracy: ", accuracy_score(y_test, preds))

!pip install koreanize-matplotlib

import koreanize_matplotlib

import matplotlib.pyplot as plt

# 상위 20개 토큰 선택
top_positive_tokens = positive_freq[:20]
top_negative_tokens = negative_freq[:20]

# 막대 그래프로 시각화
fig, ax = plt.subplots(2, 1, figsize=(10, 10))

ax[0].barh(top_positive_tokens.index, top_positive_tokens.values, color='blue')
ax[0].set_title('긍정 리뷰')

ax[1].barh(top_negative_tokens.index, top_negative_tokens.values, color='red')
ax[1].set_title('부정 리뷰')

plt.tight_layout()
plt.show()

```

MD와 동시에 구매한 금액, 건수.ipynb

▼ MD와 동시에 구매한 금액 (사용x)

```

!pip install koreanize-matplotlib

import koreanize_matplotlib

import seaborn as sns
import matplotlib.pyplot as plt
import datetime
import pandas as pd

# Load the excel file
file_path = '/content/커피템플_상세_데이터_전처리완료.xlsx'
df_sales = pd.read_excel(file_path)

# 숫자를 쉼표로 구분된 문자열로 변환하는 함수 정의
def format_number(number):
    return '{:,}'.format(int(number))

# '결제일' 열을 datetime 형식으로 변환
df_sales['결제일'] = pd.to_datetime(df_sales['결제일']) # 날짜 형식으로 변환

# '결제일'과 '결제시간'을 합쳐서 '구매시간' 열을 생성
df_sales['결제시간'] = pd.to_datetime(df_sales['결제시간'], format='%H:%M:%S').dt.time
df_sales['구매시간'] = df_sales.apply(lambda row: datetime.datetime.combine(row['결제일'], row['결제시간']), axis=1)

# MD 카테고리가 구매한 시간을 찾을
md_purchase_times = df_sales[df_sales['카테고리'] == 'MD']['구매시간']

# MD가 구매한 시간에 이루어진 모든 판매 데이터를 가져옴
df_md_related_sales = df_sales[df_sales['구매시간'].isin(md_purchase_times)]

# '년월' 열 생성
df_sales['년월'] = df_sales['결제일'].dt.to_period('M')
df_md_related_sales['년월'] = df_md_related_sales['결제일'].dt.to_period('M')

# 월별 총 판매량 계산
df_sales_monthly = df_sales.groupby('년월')['상품별 합계'].sum().reset_index()
df_md_related_monthly = df_md_related_sales.groupby('년월')['상품별 합계'].sum().reset_index()

# '년월' 열을 문자열로 변환 (그래프 생성을 위해)
df_sales_monthly['년월'] = df_sales_monthly['년월'].astype(str)
df_md_related_monthly['년월'] = df_md_related_monthly['년월'].astype(str)

# 월별 총 판매량에 대한 막대 그래프 생성
fig, ax1 = plt.subplots(figsize=(20, 10)) # 그래프의 크기를 설정
bar1 = sns.barplot(x='년월', y='상품별 합계', data=df_sales_monthly, ax=ax1, color='lightpink', label='매출액')

# 월별 MD 판매량에 대한 막대 그래프 생성
bar2 = sns.barplot(x='년월', y='상품별 합계', data=df_md_related_monthly, ax=ax1, color='deeppink', alpha=0.5, label='MD와 동시에 구매한')

# 그래프의 제목과 레이블을 설정
ax1.set_title('월별 매출액 및 MD와 동시에 구매한 금액')
ax1.set_xlabel('월')
ax1.set_ylabel('매출액')

# 텍스트 레이블 추가
for patch in bar1.patches:
    height = patch.get_height()
    ax1.text(patch.get_x()+patch.get_width()/2., height, format_number(height), color='black', ha='center', va='bottom')

for patch in bar2.patches:
    height = patch.get_height()
    ax1.text(patch.get_x()+patch.get_width()/2., height, format_number(height), color='black', ha='center', va='bottom')

plt.xticks(rotation=0)
plt.legend()
plt.show()

```

▼ 건수 시각화 (사용x)

월별 총 매출 건수 계산

```
df_sales_count_monthly = df_sales.groupby('년월').size().reset_index(name='매출 건수')
```

MD가 구매한 시간에 이루어진 월별 총 매출 건수 계산

```
df_md_related_count_monthly = df_md_related_sales.groupby('년월').size().reset_index(name='MD와 동시에 구매한 매출 건수')
```

월별 총 매출 건수에 대한 막대 그래프 생성

```

fig, ax1 = plt.subplots(figsize=(20, 10))
bar1 = sns.barplot(x='년월', y='매출 건수', data=df_sales_count_monthly, ax=ax1, color='lightblue', label='매출 건수')

# 월별 MD 매출 건수에 대한 막대 그래프 생성
bar2 = sns.barplot(x='년월', y='MD와 동시에 구매한 매출 건수', data=df_md_related_count_monthly, ax=ax1,
color='deepskyblue', alpha=0.5, label='MD와 동시에 구매한 매출 건수')

# 그래프의 제목과 레이블을 설정
ax1.set_title('월별 매출 건수 및 MD와 동시에 구매한 매출 건수')
ax1.set_xlabel('월')
ax1.set_ylabel('매출 건수')

# 텍스트 레이블 추가
for patch in bar1.patches:
    height = patch.get_height()
    ax1.text(patch.get_x()+patch.get_width()/2., height, '{:1.0f}'.format(height), color='black', ha='center', va='bottom')

for patch in bar2.patches:
    height = patch.get_height()
    ax1.text(patch.get_x()+patch.get_width()/2., height, '{:1.0f}'.format(height), color='black', ha='center', va='bottom')

plt.xticks(rotation=0)
plt.legend()
plt.show()

```

[커피템플 EDA\(정리본\).ipynb](#)

→ 요일별 매출 합계, 평균, 월별 매출액 합계, 시간대별 매출 데이터 전처리 + 시각화

▼ 요일별 매출 합계 데이터

```

import pandas as pd
import matplotlib.pyplot as plt
!pip install koreanize-matplotlib
import koreanize_matplotlib
import pandas as pd
import numpy as np

df = pd.read_excel("커피템플_상세_데이터_전처리완료.xlsx.xlsx의 사본")

# datetime으로 변환하여 분석에 용이하게 함
df['결제일'] = pd.to_datetime(df['결제일'])

# 요일 column 생성
df['요일'] = df['결제일'].dt.day_name()

result_df = df.groupby('요일').sum()

# 요일 별 매출 합계 데이터 시각화

plt.figure(figsize=(10,7))
plt.title('요일 별 매출 합계 (단위 = 원)') # 제목
plt.bar(result_df.index,result_df['상품별 합계'],color='#F1CE63', width=0.7)

# 막대그래프 위에 금액 넣기
for index, value in enumerate(result_df['상품별 합계']):
    formatted_value = "{:,.0f}".format(value) # 1000단위로 콤마 찍기
    plt.text(index, value, formatted_value, ha='center', va='bottom')

plt.show()

```

▼ 요일별 매출액 평균 데이터

```
df = pd.read_excel('커피점들_상세_데이터_전처리완료.xlsx')

# 날짜로 한 번 묶고 > 그 이후에 요일로 묶기
df['결제일'] = pd.to_datetime(df['결제일']) # datetime을 통해 결제일 처리에 용이하도록 함

df['요일'] = df['결제일'].dt.day_name() # 요일 컬럼 생성

by_date = df.groupby('결제일').sum() # groupby 결제일을 통해 결제일 별 누적 매출액 구하기

by_date['요일'] = by_date.index.day_name() # groupby(결제일(=날짜)) 한 df에 요일 컬럼 더하기

by_date_mean = by_date.groupby('요일').mean() # 요일 별 평균 구하기 mean 메서드 사용

# 요일별 매출액 평균 데이터 시각화

plt.figure(figsize=(13,7))
plt.bar(by_date_mean.index, by_date_mean['상품별 합계'], color='green', alpha=0.6)
plt.title('요일 별 매출액 평균(단위=원)')

# 막대그래프 위에 금액 쓰기
for index, value in enumerate(by_date_mean['상품별 합계']):
    formatted_value = "{:,.0f}".format(value) # 1000단위로 콤마 찍기
    plt.text(index, value+3, formatted_value, ha='center', va='bottom')
```

▼ 월별 매출액 합계

```
# 월을 추출한 column 생성
df['월'] = df['결제일'].dt.month
df['연'] = df['결제일'].dt.year

# groupby 연 & 월을 통한 데이터 확인
month_df = df.groupby(['연', '월']).sum()

month_df = month_df.reset_index() # index 초기화 - 단일 index로 변환 (groupby 2번 -> multi index)

# 월별 매출액 합계 데이터 시각화 (연-월 별 판매 합계) (2022년 2월~2023년 5월)
plt.figure(figsize=(13,7))
plt.title('연-월 별 매출액 합계(단위=원)')
plt.bar(range(len(month_df)), month_df['상품별 합계'], color='pink')
plt.plot(range(len(month_df)), month_df['상품별 합계'], color='red', linewidth=3) # 추이를 보기 위한 line graph 그리기

# 각 막대 위에 값을 표시하기 위한 반복문
for index, value in enumerate(month_df['상품별 합계']):
    formatted_value = "{:,.0f}".format(value) # 1000단위로 콤마 찍기
    plt.text(index, value, formatted_value, ha='center', va='bottom')

# x축 레이블 설정
x_labels = [f"{year}-{month}" for year, month in zip(month_df['연'], month_df['월'])]
plt.xticks(range(len(month_df)), x_labels)

plt.show()
```

▼ 시간대별 매출 데이터

```
df['시간대'] = pd.to_datetime(df['결제시간']).dt.hour

hour_sum = df.groupby('시간대')['상품별 합계'].sum()
hour_mean = df.groupby('시간대')['상품별 합계'].mean()

# 시간대별 매출 데이터 시각화

plt.figure(figsize=(13,8))
plt.bar(hour_sum.index, hour_sum)
plt.title('시간 별 매출 합계 (1시간 별, 단위=원)')
plt.xlim(8,19)

# 각 막대 위에 값을 표시하기 위한 반복문
for index, value in enumerate(hour_sum):
    formatted_value = "{:,.0f}".format(value) # 1000단위로 콤마 찍기
    plt.text(index+7, value+1, formatted_value, ha='center', va='bottom')
```


데이터분석스쿨 파이널 (1).ipynb

▼ 패키지 설치

```
!pip install koreanize-matplotlib
import koreanize_matplotlib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno # missingno : 결측데이터를 파악하는데 직관적인 도움을 주는 패키지
%matplotlib inline
import re
# 날짜컬럼 형식 바꾸기
import datetime

# 이 두 줄의 코드는 matplotlib의 기본 scheme 말고, seaborn scheme을 세팅
# 일일이 graph의 font size를 지정할 필요 없이, seaborn의 font_scale을 사용하면 편리
# plt.style.use('seaborn')
# sns.set(font_scale = 2.5)

import warnings
warnings.filterwarnings('ignore')

# 그래프에서 한글 폰트 깨지는 문제를 해결해주기 위한 코드
from matplotlib import font_manager, rc
plt.rcParams['axes.unicode_minus'] = False

import platform # 현재 운영체제를 확인

if platform.system() == 'Darwin': # macOS(Darwin)인 경우에는 rc() 함수를 사용하여 한글 폰트를 'AppleGothic'으로 설정
    rc('font', family = 'AppleGothic')
elif platform.system() == 'Windows':
    path = 'c:Windows/Fonts/Century Gothic'
    font_name = font_manager.FontProperties(fname = path).get_name()
    rc('font', family = font_name)
else:
    plt.rc('font', family='NanumGothic')
```

▼ 22년 6, 7, 8, 12월, 23년 1, 2월의 요일별 입도객 수 데이터

```
dv = pd.read_excel('요일별제주입도객.xlsx')

# '일자' 열을 활용하여 요일을 추출하여 '요일', '월', '일' 열을 추가
dv['요일'] = dv['일자'].dt.day_name()
dv['연'] = dv['일자'].dt.year
dv['월'] = dv['일자'].dt.month

# 요일 한국어로 저장
weekday_korean = {
    'Monday': '월요일',
    'Tuesday': '화요일',
    'Wednesday': '수요일',
    'Thursday': '목요일',
    'Friday': '금요일',
    'Saturday': '토요일',
    'Sunday': '일요일'
}

dv['요일'] = dv['요일'].map(weekday_korean)

# 22년 6, 7, 8, 12월 23년 1, 2월 입도객 수 그래프 그리기
months = dv['월'].unique()

for month in months:
    data_month = dv[dv['월'] == month]
    pivot_table = data_month.pivot_table(index='요일', values='일별 입도객', aggfunc='sum')

    pivot_table = pivot_table.sort_values(by='일별 입도객', ascending=False)

    plt.figure(figsize=(10, 6))
    plt.bar(pivot_table.index, pivot_table['일별 입도객'], color='#F1CE63', width=0.7)

    # 요일의 첫 글자로 xlabel 설정
    plt.xlabel('요일', fontsize=10)
```

```

plt.xticks(range(len(pivot_table.index)), [day[0] for day in pivot_table.index], fontsize=10) # 첫 글자만 사용

plt.ylabel('입도객 수', fontsize=10)
plt.title(f'{month}월 요일별 입도객 수(합계)', fontsize=10)

# 막대 위에 값을 표시
for index, value in enumerate(pivot_table['일별 입도객']):
    plt.annotate(f'{value}', xy=(index, value), ha='center', va='bottom', fontsize=10)

plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

for spine in plt.gca().spines.values():
    spine.set_color('black')
plt.gca().set_facecolor('white')

plt.show()

```

▼ 월별 주문수 상위 10개 상품 데이터

```

ctd = pd.read_excel('커피템플_상세_데이터_전처리완료 (1).xlsx')

# 결제일, 결제시간이 문자형이니까 먼저 결제일+결제시간 만들기
ctd['결제ID'] = ctd['결제일'] + " " + ctd['결제시간']

# 결제일, 시간을 날짜형식으로 바꾸기
ctd['결제일'] = pd.to_datetime(ctd['결제일'], format = "%Y-%m-%d", errors = 'raise') # 에러발생 -> 에러발생
ctd['결제ID'] = pd.to_datetime(ctd['결제ID'], format="%Y-%m-%d %H:%M:%S", errors='raise')

# 연, 월 컬럼 생성
ctd['연'] = ctd['결제일'].dt.year
ctd['월'] = ctd['결제일'].dt.month

# '결제일' 열을 활용하여 요일을 추출하여 '요일' 열을 추가
ctd['요일'] = ctd['결제일'].dt.day_name()

# 필요없는 상품명 제거
# 정규표현식 패턴 설정
pattern = r'벨\d+'

# 여러 조건을 포함하여 필터링(더 있음)
ctd_n = ctd[
    (~ctd['상품명'].str.contains('종이백')) &
    (~ctd['상품명'].str.contains('창가')) &
    (~ctd['상품명'].str.contains('큰테이블')) &
    (~ctd['상품명'].str.contains('연하계')) &
    (~ctd['상품명'].str.contains(pattern, regex=True)) &
    (~ctd['상품명'].str.contains('포장')) &
    (~ctd['상품명'].str.contains('아외')) &
    (~ctd['상품명'].str.contains('적게')) &
    (~ctd['상품명'].str.contains('추가')) &
    (~ctd['상품명'].str.contains('탈개')) &
    (~ctd['상품명'].str.contains('보냉백'))
]

# 2022년도 월별 상품 주문수 상위 10개를 저장
ctd_dict = {}

for i in range(2, 13):
    var_name = f'ctd_22{i}'
    ctd_dict[var_name] = ctd_n[(ctd_n['연'] == 2022) & (ctd_n['월'] == i)]['상품명'].value_counts().reset_index().head(10)
    ctd_dict[var_name].columns = [f'{i}월_상품명', f'{i}월_주문수']

# 2023년도 월별 상품 주문수 상위 10개를 저장
ctd_dict1 = {}

for i in range(1, 6):
    var_name = f'ctd_23{i}'
    ctd_dict1[var_name] = ctd_n[(ctd_n['연'] == 2023) & (ctd_n['월'] == i)]['상품명'].value_counts().reset_index().head(10)
    ctd_dict1[var_name].columns = [f'{i}월_상품명', f'{i}월_주문수']

# 22년 각 월별 상위 10개 상품별 주문수 그래프 그리기
for i in range(2, 13):
    var_name = f'ctd_22{i}'
    sorted_data = ctd_dict[var_name].sort_values(by=f'{i}월_주문수', ascending=True)
    plt.figure(figsize=(10, 6))
    plt.barh(sorted_data[f'{i}월_상품명'], sorted_data[f'{i}월_주문수'], color='#F1CE63') # x-axis and y-axis are swapped
    plt.xlabel('주문수', fontsize=10)
    plt.ylabel('상품명', fontsize=10)
    plt.title(f'22년 {i}월 주문수 상위 10개 상품명', fontsize=10) # title

```

```

# 막대 위에 값을 표시
for index, value in enumerate(sorted_data[f'{i}월_주문수']):
    plt.text(value, index, str(value), ha='left', va='center', fontsize=8)

plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

for spine in plt.gca().spines.values():
    spine.set_color('black')
plt.gca().set_facecolor('white')

plt.show()

# 23년 각 월별 상위 10개 상품별 주문수 그래프 그리기
for i in range(1, 6):
    var_name = f'ctd_23{i}'
    sorted_data = ctd_dict1[var_name].sort_values(by=f'{i}월_주문수', ascending=True) # 오름차순으로 정렬
    plt.figure(figsize=(10, 6))
    plt.barh(sorted_data[f'{i}월_상품명'], sorted_data[f'{i}월_주문수'], color='#F1CE63') # x-axis and y-axis are swapped
    plt.xlabel('주문수', fontsize=10) # x-axis label
    plt.ylabel('상품명', fontsize=10) # y-axis label
    plt.title(f'23년 {i}월 주문수 상위 10개 상품명', fontsize=10) # title
    plt.tight_layout()

    plt.xticks(fontsize=8) # x축 눈금 레이블 글씨 크기 조정
    plt.yticks(fontsize=10) # y축 눈금 레이블 글씨 크기 조정

    for spine in plt.gca().spines.values():
        spine.set_color('black')
    plt.gca().set_facecolor('white')

# 막대 위에 값을 표시하는 로직 추가
for index, value in enumerate(sorted_data[f'{i}월_주문수']):
    plt.text(value, index, str(value), ha='left', va='center', fontsize=8)

plt.show()

# 차트레이스
!pip install bar_chart_race
import bar_chart_race as bcr

data = '데이터 비공개'

# '날짜' 열을 인덱스로 설정합니다
data.set_index('날짜', inplace=True)

# 애니메이션 라인 차트 레이스를 생성합니다
bcr.bar_chart_race(data, n_bars=10, steps_per_period=10)

```

▼ 시간대별 주문수 상위 10개 상품 데이터

```

ctd = pd.read_excel('커피템플_상세_데이터_전처리완료 (1).xlsx')

# 결제일, 결제시간이 문자형이니까 먼저 결제일+결제시간 만들기
ctd['결제ID'] = ctd['결제일'] + " " + ctd['결제시간']

# 결제일, 시간을 날짜형식으로 바꾸기
ctd['결제일'] = pd.to_datetime(ctd['결제일'], format = "%Y-%m-%d", errors = 'raise') # 에러발생 -> 에러발생
ctd['결제ID'] = pd.to_datetime(ctd['결제ID'], format="%Y-%m-%d %H:%M:%S", errors='raise')

# 연, 월 컬럼 생성
ctd['연'] = ctd['결제일'].dt.year
ctd['월'] = ctd['결제일'].dt.month

# '결제일' 열을 활용하여 요일을 추출하여 '요일' 열을 추가
ctd['요일'] = ctd['결제일'].dt.day_name()

# 필요없는 상품명 제거
# 정규표현식 패턴 설정
pattern = r'벨\d+'

# 여러 조건을 포함하여 필터링(더 있음)
ctd_n = ctd[
    (~ctd['상품명'].str.contains('종이백')) &
    (~ctd['상품명'].str.contains('차가')) &
    (~ctd['상품명'].str.contains('콘테이블')) &
    (~ctd['상품명'].str.contains('연하계')) &
    (~ctd['상품명'].str.contains(pattern, regex=True)) &
    (~ctd['상품명'].str.contains('포장')) &
    (~ctd['상품명'].str.contains('야외')) &
    (~ctd['상품명'].str.contains('적게')) &

```

```

(~ctd['상품명'].str.contains('추가')) &
(~ctd['상품명'].str.contains('달게')) &
(~ctd['상품명'].str.contains('보냉백'))
]

# 시간별 상품 주문수 상위 10개를 저장 (23년 4,5월 메뉴 이름 바뀌어 때문에 전처리 다시해야됨)
ctd_dict2 = {}

for i in range(7, 20):
    var_name = f'ctd_h_{i}'
    ctd_dict2[var_name] = ctd_n[(ctd_n['결제ID'].dt.hour == i)][['상품명']].value_counts().reset_index().head(10)
    ctd_dict2[var_name].columns = [f'{i}시_상품명', f'{i}시_주문수']

# 각 시간별 상위 10개 상품별 주문수 그래프 그리기
for i in range(7, 19):
    var_name = f'ctd_h_{i}'
    sorted_data = ctd_dict2[var_name].sort_values(by=f'{i}시_주문수', ascending=True) # 오름차순으로 정렬
    plt.figure(figsize=(10, 6))
    plt.barh(sorted_data[f'{i}시_상품명'], sorted_data[f'{i}시_주문수'], color='#F1CE63') # x-axis and y-axis are swapped
    plt.xlabel('주문수', fontsize=10) # x-axis label
    plt.ylabel('상품명', fontsize=10) # y-axis label
    plt.title(f'{i}시 주문수 상위 10개 상품명', fontsize=10) # title
    plt.tight_layout()

    plt.xticks(fontsize=8) # x축 눈금 레이블 글씨 크기 조정
    plt.yticks(fontsize=10) # y축 눈금 레이블 글씨 크기 조정

    for spine in plt.gca().spines.values():
        spine.set_color('black')
    plt.gca().set_facecolor('white')

    # 막대 위에 값을 표시하는 로직 추가
    for index, value in enumerate(sorted_data[f'{i}시_주문수']):
        plt.text(value, index, str(value), ha='left', va='center', fontsize=8)

plt.show()

```

▼ 월별 시간대별 매출

```

ctd = pd.read_excel('커피점포_상세_데이터_전처리완료 (1).xlsx')

# 결제일, 결제시간이 문자형이니까 먼저 결제일+결제시간 만들기
ctd['결제ID'] = ctd['결제일'] + " " + ctd['결제시간']

# 결제일, 시간을 날짜형식으로 바꾸기
ctd['결제일'] = pd.to_datetime(ctd['결제일'], format = "%Y-%m-%d", errors = 'raise') # 에러발생 -> 에러발생
ctd['결제ID'] = pd.to_datetime(ctd['결제ID'], format="%Y-%m-%d %H:%M:%S", errors='raise')

# 연, 월 컬럼 생성
ctd['연'] = ctd['결제일'].dt.year
ctd['월'] = ctd['결제일'].dt.month

# '결제일' 열을 활용하여 요일을 추출하여 '요일' 열을 추가
ctd['요일'] = ctd['결제일'].dt.day_name()

# 필요없는 상품명 제거
# 정규표현식 패턴 설정
pattern = r'별\d+'

# 여러 조건을 포함하여 필터링(더 있음)
ctd_n = ctd[
    (~ctd['상품명'].str.contains('종이백')) &
    (~ctd['상품명'].str.contains('창가')) &
    (~ctd['상품명'].str.contains('큰테이블')) &
    (~ctd['상품명'].str.contains('연하게')) &
    (~ctd['상품명'].str.contains(pattern, regex=True)) &
    (~ctd['상품명'].str.contains('포장')) &
    (~ctd['상품명'].str.contains('아외')) &
    (~ctd['상품명'].str.contains('적게')) &
    (~ctd['상품명'].str.contains('추가')) &
    (~ctd['상품명'].str.contains('달게')) &
    (~ctd['상품명'].str.contains('보냉백'))
]

# 2022년도 월별 시간대별 매출
ctd_dict3 = {}

for i in range(2, 13):
    var_name = f'ctd_22s_{i}'
    # '결제ID' 컬럼의 데이터를 datetime 형식으로 변환하여 시간 정보 추출 후 그룹화
    ctd_n['결제ID'] = pd.to_datetime(ctd_n['결제ID'])
    monthly_sales = ctd_n[(ctd_n['연'] == 2022) & (ctd_n['월'] == i)].groupby(ctd_n['결제ID'].dt.hour)['상품별 합계'].sum()

```

```

# 시간과 매출 컬럼으로 데이터프레임 생성 후 딕셔너리에 저장
ctd_dict3[var_name] = pd.DataFrame({'시간': monthly_sales.index, '매출': monthly_sales.values})

# 2023년도 월별 시간대별 매출합계
ctd_dict4 = {}

for i in range(1, 6):
    var_name = f'ctd_23s{i}'
    # '결제ID' 컬럼의 데이터를 datetime 형식으로 변환하여 시간 정보 추출 후 그룹화
    ctd_n['결제ID'] = pd.to_datetime(ctd_n['결제ID'])
    monthly_sales = ctd_n[(ctd_n['연'] == 2023) & (ctd_n['월'] == i)].groupby(ctd_n['결제ID'].dt.hour)['상품별 합계'].sum()
    # 시간과 매출 컬럼으로 데이터프레임 생성 후 딕셔너리에 저장
    ctd_dict4[var_name] = pd.DataFrame({'시간': monthly_sales.index, '매출': monthly_sales.values})

# 22년 시간대별 매출
for i in range(2, 13):
    var_name = f'ctd_22s{i}'
    sorted_data = ctd_dict3[var_name].sort_values(by=['시간'], ascending=True)
    plt.figure(figsize=(10, 6))
    plt.bar(sorted_data['시간'], sorted_data['매출'], color='#F1CE63', width=0.7)
    plt.xlabel('시간', fontsize=10)
    plt.ylabel('매출', fontsize=10)
    plt.title(f'22년 {i}월 시간대별 매출', fontsize=10)
    plt.ticklabel_format(style='plain')

    for index, value in enumerate(sorted_data['매출']):
        plt.text(sorted_data['시간'].iloc[index], value, str(value), ha='center', va='bottom')

    plt.xticks(fontsize=10)
    plt.yticks(fontsize=10)

    for spine in plt.gca().spines.values():
        spine.set_color('black')
    plt.gca().set_facecolor('white')

    plt.show()

# 23년 시간대별 매출
for i in range(1, 6):
    var_name = f'ctd_23s{i}'
    sorted_data = ctd_dict4[var_name].sort_values(by=['시간'], ascending=True)
    plt.figure(figsize=(10, 6))
    plt.bar(sorted_data['시간'], sorted_data['매출'], color='#F1CE63', width=0.7)
    plt.xlabel('시간', fontsize=10)
    plt.ylabel('매출', fontsize=10)
    plt.title(f'23년 {i}월 시간대별 매출', fontsize=10)
    plt.ticklabel_format(style='plain')

    # 막대 위에 값을 표시하는 로직 추가
    for index, value in enumerate(sorted_data['매출']):
        plt.text(sorted_data['시간'].iloc[index], value, str(value), ha='center', va='bottom')

    plt.xticks(fontsize=10)
    plt.yticks(fontsize=10)

    # Set black border with a white background
    for spine in plt.gca().spines.values():
        spine.set_color('black')
    plt.gca().set_facecolor('white')

    plt.show()

```

▼ 월별 매출과 관광객 수 데이터

```

ctd = pd.read_excel('커피점별 상세 데이터 전처리완료 (1).xlsx')
visitor = pd.read_excel('22.2~23.5 관광객입도현황.xlsx')

ctd['결제ID'] = ctd['결제일'] + " " + ctd['결제시간']
ctd['결제일'] = pd.to_datetime(ctd['결제일'], format="%Y-%m-%d", errors='raise') # 에러발생 -> 에러발생
ctd['결제ID'] = pd.to_datetime(ctd['결제ID'], format="%Y-%m-%d %H:%M:%S", errors='raise')
ctd['연'] = ctd['결제일'].dt.year
ctd['월'] = ctd['결제일'].dt.month
ctd['요일'] = ctd['결제일'].dt.day_name()

visitor['연'] = visitor['날짜'].dt.year
visitor['월'] = visitor['날짜'].dt.month
visitor_s = visitor[['연', '월', '총계']]
visitor_s = visitor_s.reset_index()
visitor_s.rename(columns={'총계': '관광객'}, inplace=True)

yms_v = pd.merge(ctd_yms, visitor_s, on=['연', '월'])
yms_v = yms_v.drop('index', axis=1)

```

```

yms_v['날짜'] = yms_v['연'].astype(str) + '-' + yms_v['월'].astype(str)
yms_v['날짜'] = pd.to_datetime(yms_v['날짜'], format='%Y-%m')
yms_v = yms_v.reset_index()
yms_v = yms_v.drop('index', axis=1)

# 날짜별 매출과 관광객 그래프

# 이중 축을 가진 그림과 서브플롯 생성
fig, ax1 = plt.subplots(figsize=(12, 8)) # figsize로 그래프의 크기 조정

# 첫 번째 축에 날짜별 매출 그래프 그리기
ax1.plot(yms_v['날짜'].dt.strftime('%Y/%m'), yms_v['매출'], label='날짜별 매출', marker='o', color='tab:red')
ax1.set_xlabel('날짜', fontsize=14) # fontsize로 x축 레이블의 크기 조정
ax1.set_ylabel('매출', color='tab:red', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax1.tick_params(axis='y', labelcolor='tab:red', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# 두 번째 축 생성
ax2 = ax1.twinx()

# 두 번째 축에 날짜별 관광객 그래프 그리기
ax2.plot(yms_v['날짜'].dt.strftime('%Y/%m'), yms_v['관광객'], label='날짜별 관광객', marker='x', color='tab:blue')
ax2.set_ylabel('관광객 수', color='tab:blue', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax2.tick_params(axis='y', labelcolor='tab:blue', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# y축 격자 표시
ax2.grid(axis='y')

# x축 눈금 표시 간격 설정 (1달 간격으로 눈금 표시) ???
ax1.tick_params(axis='x', rotation=45, labelsz=12) # fontsize로 x축 눈금의 크기 조정

# x축 눈금의 크기를 8개로 제한
ax1.xaxis.set_major_locator(plt.MaxNLocator(10))

# 그래프 제목과 범례 설정
plt.title('날짜별 매출과 관광객 수', fontsize=16) # fontsize로 제목의 크기 조정

# 격자 표시
ax1.grid(True)

plt.tight_layout()
plt.show()

```

▼ 월별 주문수와 관광객 수 데이터

```

ctd = pd.read_excel('커피점들_상세_데이터_전처리완료 (1).xlsx')
visitor = pd.read_excel('22.2~23.5 관광객입도현황.xlsx')

ctd['결제ID'] = ctd['결제일'] + " " + ctd['결제시간']
ctd['결제일'] = pd.to_datetime(ctd['결제일'], format = "%Y-%m-%d", errors = 'raise') # 에러발생 -> 에러발생
ctd['결제ID'] = pd.to_datetime(ctd['결제ID'], format="%Y-%m-%d %H:%M:%S", errors='raise')
ctd['연'] = ctd['결제일'].dt.year
ctd['월'] = ctd['결제일'].dt.month
ctd['요일'] = ctd['결제일'].dt.day_name()

visitor['연'] = visitor['날짜'].dt.year
visitor['월'] = visitor['날짜'].dt.month
visitor_s = visitor[['연', '월', '총계']]
visitor_s = visitor_s.reset_index()
visitor_s.rename(columns={'총계': '관광객'}, inplace=True)

ymoc_v = pd.merge(ctd_oc, visitor_s, on=['연', '월'])
ymoc_v = ymoc_v.drop('index', axis=1)
ymoc_v['날짜'] = ymoc_v['연'].astype(str) + '-' + ymoc_v['월'].astype(str)
ymoc_v['날짜'] = pd.to_datetime(ymoc_v['날짜'], format='%Y-%m')
ymoc_v = ymoc_v.reset_index()
ymoc_v = ymoc_v.drop('index', axis=1)

# 날짜별 주문건수와 관광객 그래프

# 이중 축을 가진 그림과 서브플롯 생성
fig, ax1 = plt.subplots(figsize=(12, 8)) # figsize로 그래프의 크기 조정

# 첫 번째 축에 날짜별 주문수 그래프 그리기
ax1.plot(ymoc_v['날짜'].dt.strftime('%Y/%m'), ymoc_v['주문수'], label='날짜별 주문수', marker='o', color='tab:red')
ax1.set_xlabel('날짜', fontsize=14) # fontsize로 x축 레이블의 크기 조정
ax1.set_ylabel('주문수', color='tab:red', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax1.tick_params(axis='y', labelcolor='tab:red', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# 두 번째 축 생성
ax2 = ax1.twinx()

```

```
# 두 번째 축에 날짜별 관광객 그래프 그리기
ax2.plot(ymoc_v['날짜'].dt.strftime('%y/%m'), ymoc_v['관광객'], label='날짜별 관광객', marker='x', color='tab:blue')
ax2.set_ylabel('관광객 수', color='tab:blue', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax2.tick_params(axis='y', labelcolor='tab:blue', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# y축 격자 표시
ax2.grid(axis='y')

# x축 눈금 표시 간격 설정 (1달 간격으로 눈금 표시) ???
ax1.tick_params(axis='x', rotation=45, labelsz=12) # fontsize로 x축 눈금의 크기 조정

# x축 눈금의 크기를 8개로 제한
ax1.xaxis.set_major_locator(plt.MaxNLocator(10))

# 그래프 제목과 범례 설정
plt.title('날짜별 방문수와 관광객 수', fontsize=16) # fontsize로 제목의 크기 조정

# 격자 표시
ax1.grid(True)

plt.tight_layout()
plt.show()
```

▼ 해외관광객, 제주관광객 수 데이터 (사용x)

```
visitor = pd.read_excel('22.2~23.5 관광객입도현황.xlsx')
ta = pd.read_excel('해외출국현황.xls')

visitor['연'] = visitor['날짜'].dt.year
visitor['월'] = visitor['날짜'].dt.month
visitor_s = visitor[['연', '월', '총계']]
visitor_s = visitor_s.reset_index()
visitor_s.rename(columns={'총계': '관광객'}, inplace=True)

ta.rename(columns={'전체': '날짜', '인원': '해외관광객'}, inplace=True)
ta['날짜'] = pd.to_datetime(ta['날짜'], format='%Y년%m월')
ta['연'] = ta['날짜'].dt.year
ta['월'] = ta['날짜'].dt.month

v_ta = pd.merge(ta, visitor_s, on=['연', '월'])
v_ta = v_ta.drop('index', axis=1)

# 해외관광객 제주관광객 수 비교 그래프

# 이중 축을 가진 그림과 서브플롯 생성
fig, ax1 = plt.subplots(figsize=(12, 8)) # figsize로 그래프의 크기 조정

# 첫 번째 축에 날짜별 해외 그래프 그리기
ax1.plot(v_ta['날짜'].dt.strftime('%y/%m'), v_ta['해외관광객'], label='날짜별 방문수', marker='o', color='tab:red')
ax1.set_xlabel('날짜', fontsize=14) # fontsize로 x축 레이블의 크기 조정
ax1.set_ylabel('해외관광객 수', color='tab:red', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax1.tick_params(axis='y', labelcolor='tab:red', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# 두 번째 축 생성
ax2 = ax1.twinx()

# 두 번째 축에 날짜별 관광객 그래프 그리기
ax2.plot(v_ta['날짜'].dt.strftime('%y/%m'), v_ta['관광객'], label='날짜별 관광객', marker='x', color='tab:blue')
ax2.set_ylabel('관광객 수', color='tab:blue', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax2.tick_params(axis='y', labelcolor='tab:blue', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# y축 격자 표시
ax2.grid(axis='y')

# x축 눈금 표시 간격 설정 (1달 간격으로 눈금 표시) ???
ax1.tick_params(axis='x', rotation=45, labelsz=12) # fontsize로 x축 눈금의 크기 조정

# x축 눈금의 크기를 8개로 제한
ax1.xaxis.set_major_locator(plt.MaxNLocator(10))

# 그래프 제목과 범례 설정
plt.title('월별 해외관광객 수 와 제주관광객 수', fontsize=16) # fontsize로 제목의 크기 조정

# 격자 표시
ax1.grid(True)

plt.tight_layout()
plt.show()
```

▼ 월별 매출과 관광객의 전체업종 소비금액 (사용x)

```

ctd = pd.read_excel('커피템플_상세_데이터_전처리완료 (1).xlsx')
pc = pd.read_excel('2202-2305_관광소비 추이_내국인.xlsx')

ctd['결제ID'] = ctd['결제일'] + " " + ctd['결제시간']
ctd['결제일'] = pd.to_datetime(ctd['결제일'], format = "%Y-%m-%d", errors = 'raise')
ctd['결제ID'] = pd.to_datetime(ctd['결제ID'], format = "%Y-%m-%d %H:%M:%S", errors = 'raise')
ctd['연'] = ctd['결제일'].dt.year
ctd['월'] = ctd['결제일'].dt.month
ctd['요일'] = ctd['결제일'].dt.day_name()

pc['연'] = pc['기준연월'].dt.year
pc['월'] = pc['기준연월'].dt.month
yms_pc = pd.merge(ctd_yms, pc, on=['연', '월'])

# 매출과 제주관광객의 소비 비교

# 이중 축을 가진 그림과 서브플롯 생성
fig, ax1 = plt.subplots(figsize=(12, 8)) # figsize로 그래프의 크기 조정

# 첫 번째 축에 월별 매출 그리기
ax1.plot(yms_pc['기준연월'].dt.strftime('%y/%m'), yms_pc['매출'], label='월별 매출', marker='o', color='tab:red')
ax1.set_xlabel('기준연월', fontsize=14) # fontsize로 x축 레이블의 크기 조정
ax1.set_ylabel('매출', color='tab:red', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax1.tick_params(axis='y', labelcolor='tab:red', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# 두 번째 축 생성
ax2 = ax1.twinx()

# 두 번째 축에 월별 제주관광객 전체업종소비액 그래프 그리기
ax2.plot(yms_pc['기준연월'].dt.strftime('%y/%m'), yms_pc['전체업종소비액'], label='월별 소비금액', marker='x', color='tab:blue')
ax2.set_ylabel('전체업종소비금액', color='tab:blue', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax2.tick_params(axis='y', labelcolor='tab:blue', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# y축 격자 표시
ax2.grid(axis='y')

# x축 눈금 표시 간격 설정 (1달 간격으로 눈금 표시) ???
ax1.tick_params(axis='x', rotation=45, labelsz=12) # fontsize로 x축 눈금의 크기 조정

# x축 눈금의 크기를 8개로 제한
ax1.xaxis.set_major_locator(plt.MaxNLocator(10))

# 그래프 제목과 범례 설정
plt.title('월별 매출과 제주관광객 전체업종 소비금액', fontsize=16) # fontsize로 제목의 크기 조정

# 격자 표시
ax1.grid(True)

plt.tight_layout()
plt.show()

```

▼ 월별 매출과 관광객의 식음료 업종 소비금액 (사용x)

```

ctd = pd.read_excel('커피템플_상세_데이터_전처리완료 (1).xlsx')
pc = pd.read_excel('2202-2305_관광소비 추이_내국인.xlsx')

ctd['결제ID'] = ctd['결제일'] + " " + ctd['결제시간']
ctd['결제일'] = pd.to_datetime(ctd['결제일'], format = "%Y-%m-%d", errors = 'raise')
ctd['결제ID'] = pd.to_datetime(ctd['결제ID'], format = "%Y-%m-%d %H:%M:%S", errors = 'raise')
ctd['연'] = ctd['결제일'].dt.year
ctd['월'] = ctd['결제일'].dt.month
ctd['요일'] = ctd['결제일'].dt.day_name()

pc['연'] = pc['기준연월'].dt.year
pc['월'] = pc['기준연월'].dt.month
yms_pc = pd.merge(ctd_yms, pc, on=['연', '월'])

# 매출과 제주관광객의 소비 비교

# 이중 축을 가진 그림과 서브플롯 생성
fig, ax1 = plt.subplots(figsize=(12, 8)) # figsize로 그래프의 크기 조정

# 첫 번째 축에 월별 매출 그리기
ax1.plot(yms_pc['기준연월'].dt.strftime('%y/%m'), yms_pc['매출'], label='월별 매출', marker='o', color='tab:red')
ax1.set_xlabel('기준연월', fontsize=14) # fontsize로 x축 레이블의 크기 조정
ax1.set_ylabel('매출', color='tab:red', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax1.tick_params(axis='y', labelcolor='tab:red', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# 두 번째 축 생성
ax2 = ax1.twinx()

```



```

# 두 번째 축에 월별 제주관광객 식음료업소비액 그래프 그리기
ax2.plot(yms_pc['기준연월'].dt.strftime('%y/%m'), yms_pc['식음료업소비액'], label='월별 소비금액', marker='x', color='tab:blue')
ax2.set_ylabel('전체업종소비금액', color='tab:blue', fontsize=14) # fontsize로 y축 레이블의 크기 조정
ax2.tick_params(axis='y', labelcolor='tab:blue', labelsz=12) # labelsz로 y축 눈금의 크기 조정

# y축 격자 표시
ax2.grid(axis='y')

# x축 눈금 표시 간격 설정 (1달 간격으로 눈금 표시) ???
ax1.tick_params(axis='x', rotation=45, labelsz=12) # fontsize로 x축 눈금의 크기 조정

# x축 눈금의 크기를 8개로 제한
ax1.xaxis.set_major_locator(plt.MaxNLocator(10))

# 그래프 제목과 범례 설정
plt.title('월별 매출과 제주관광객 식음료업 소비금액', fontsize=16) # fontsize로 제목의 크기 조정

# 격자 표시
ax1.grid(True)

plt.tight_layout()
plt.show()

```

3-4. 활용 라이브러리 등 기술적 요소

- ☑ 주파타
- ☑ 태블로
- ☑ 코랩
- ☑ 파크마

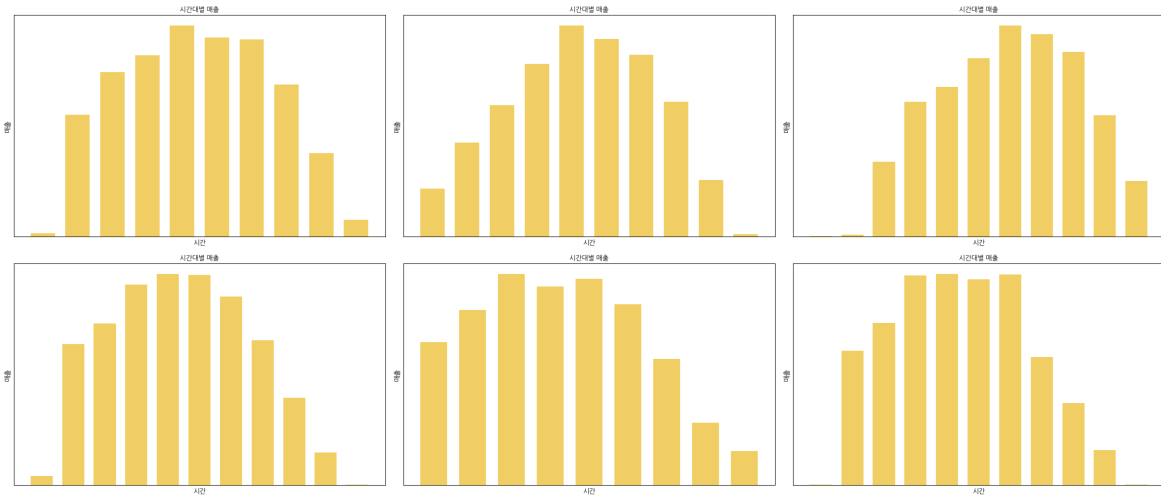
3-5. 프로젝트에서 분석한 내용

- ☑ 월, 일, 요일, 시간대별 매출
- ☑ 날씨(기온, 강수량)에 따른 매출
- ☑ 임도객(관광객) 수 관련 매출, 주문수
- ☑ 평일, 휴일, 공휴일의 매출 차이
- ☑ MD 매출 (MD와 함께 주문한 상품 포함)
- ☑ 관광객 소비금액과 매출의 경향
- ☑ 월, 시간대별 판매건수 상위 메뉴
- ☑ 해외관광객, 제주관광객 수의 변화

3-6. 결과물 상세

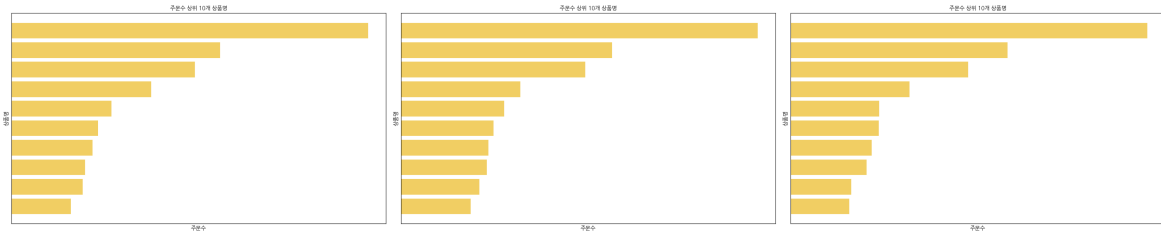
▼ 데이터 전처리를 통한 그래프

- 여름(6, 7, 8월), 겨울(12, 1, 2월) 시간대별 매출

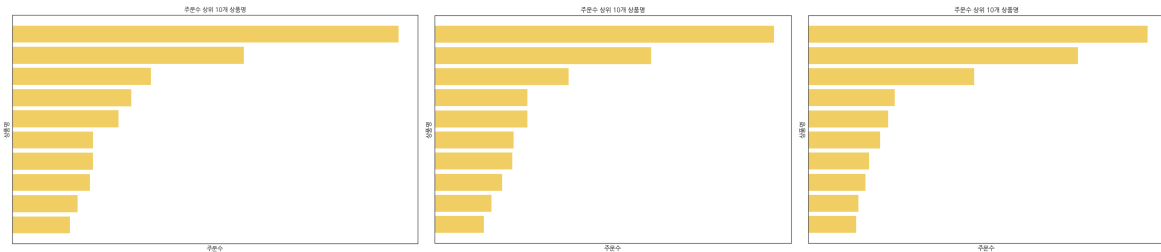


→ 시간대별 매출을 파악하여 클래스 운영시간 선정에 도움을 주는 그래프

- 여름(6, 7, 8월) 주문수 상위 10개 품목



- 겨울(12, 1, 2월) 주문수 상위 10개 품목



→ 여름, 겨울의 상위 10개 품목을 뽑아 여름과 겨울의 인기 품목을 파악 및 클래스 메뉴 선택에 도움을 주는 그래프

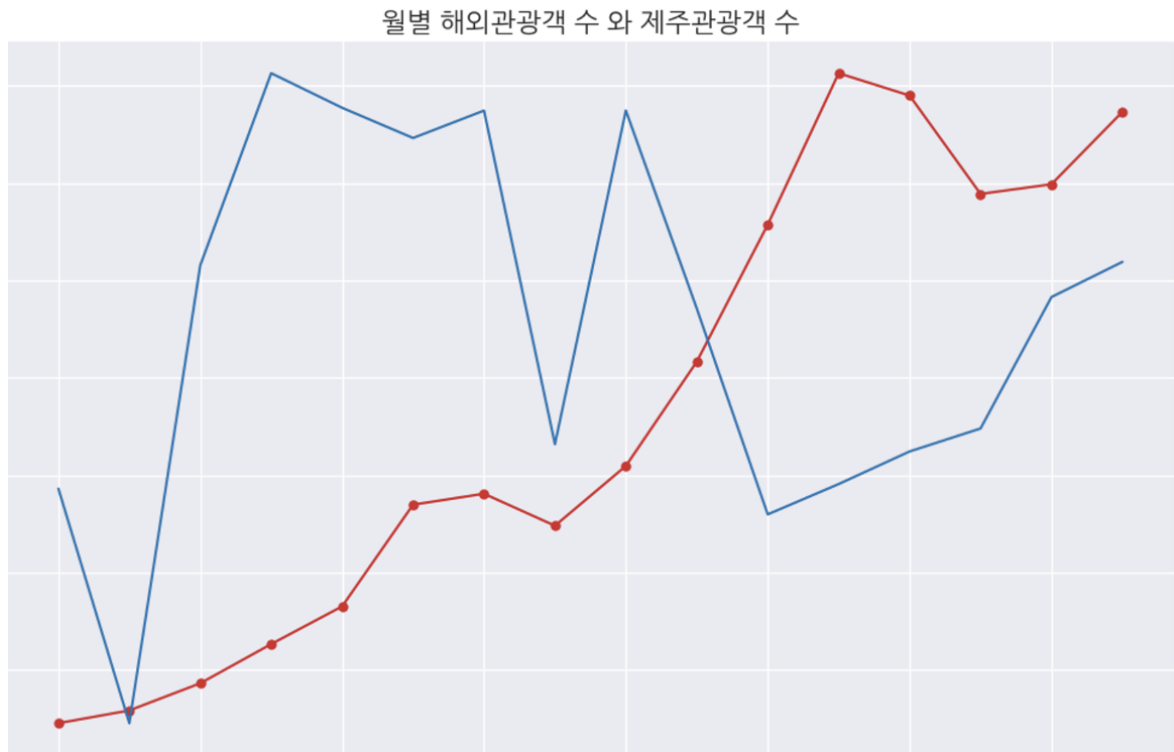
- 날짜별 매출과 관광객 수 / 주문수와 관광객 수



→ 매출, 주문수와 관광객 수의 관계를 파악 할 수 있는 그래프

관광객 수의 증가가 매출 증가와 영향이 없음을 보이기 때문에 커피템플을 방문 해야 하는 이유(클래스)를 만들어야겠다고 생각, 클래스 운영을 통해 떨어진 매출을 회복 할 수 있을 것으로 판단

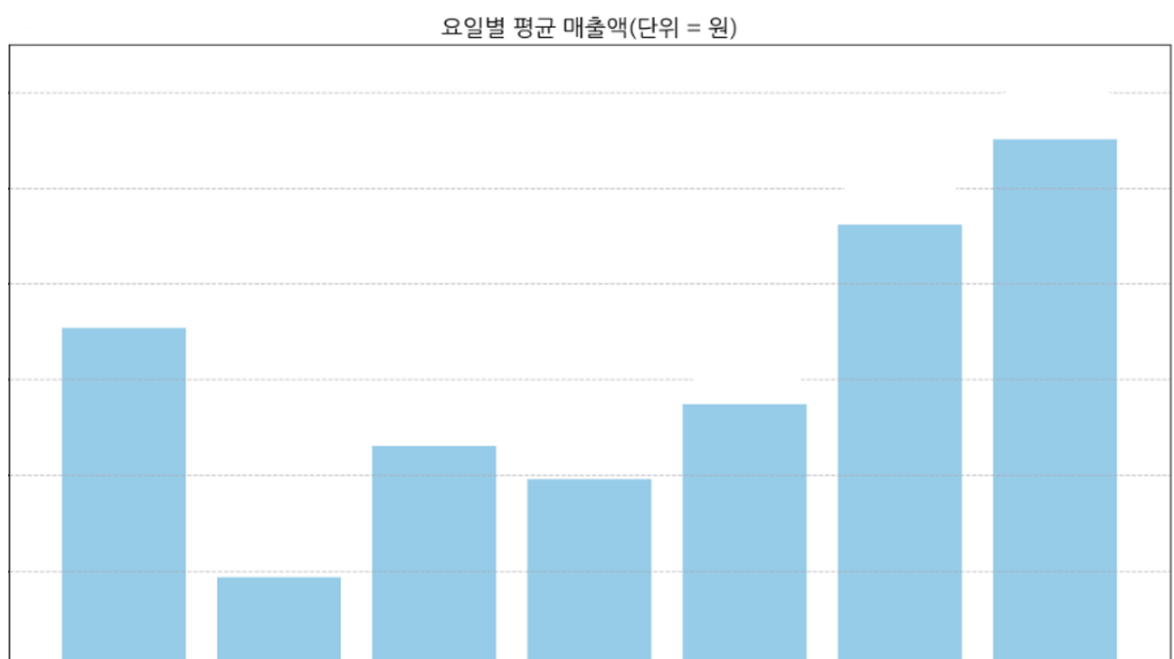
- 월별 해외관광객 수와 제주관광객 수



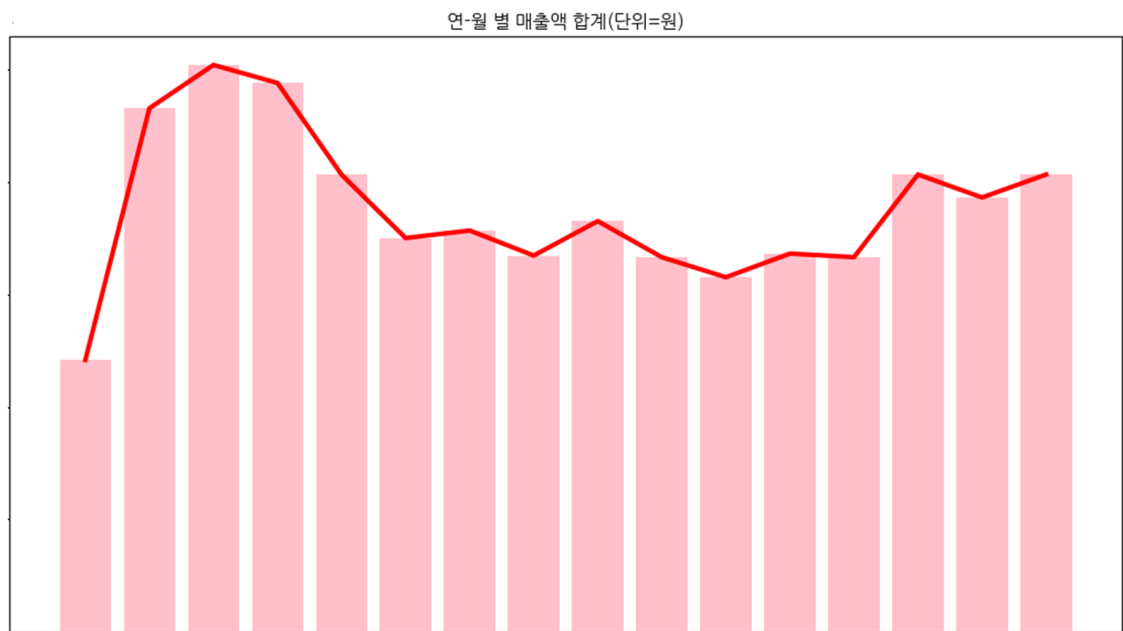
→ 해외 관광객 수와 제주관광객 수의 상관 관계를 파악 할 수 있는 그래프

해외 관광객 수는 점점 증가 할 수록 제주 관광객은 비교적 줄어들

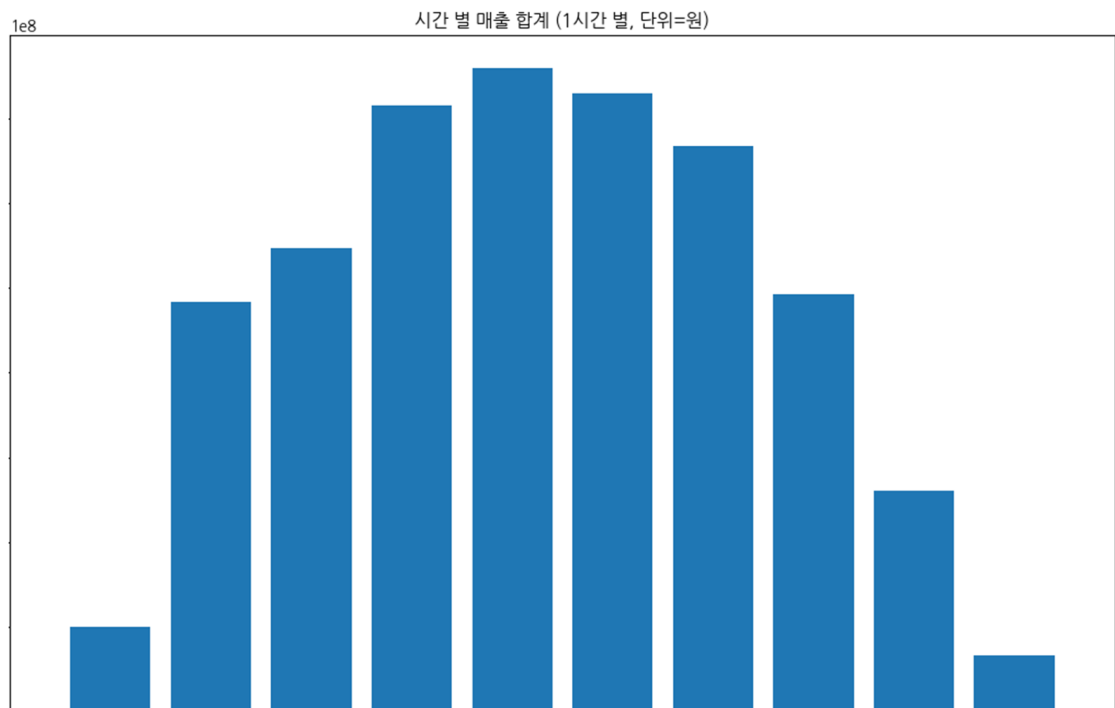
- 요일별 매출액 평균



- 연-월별 매출액 합계

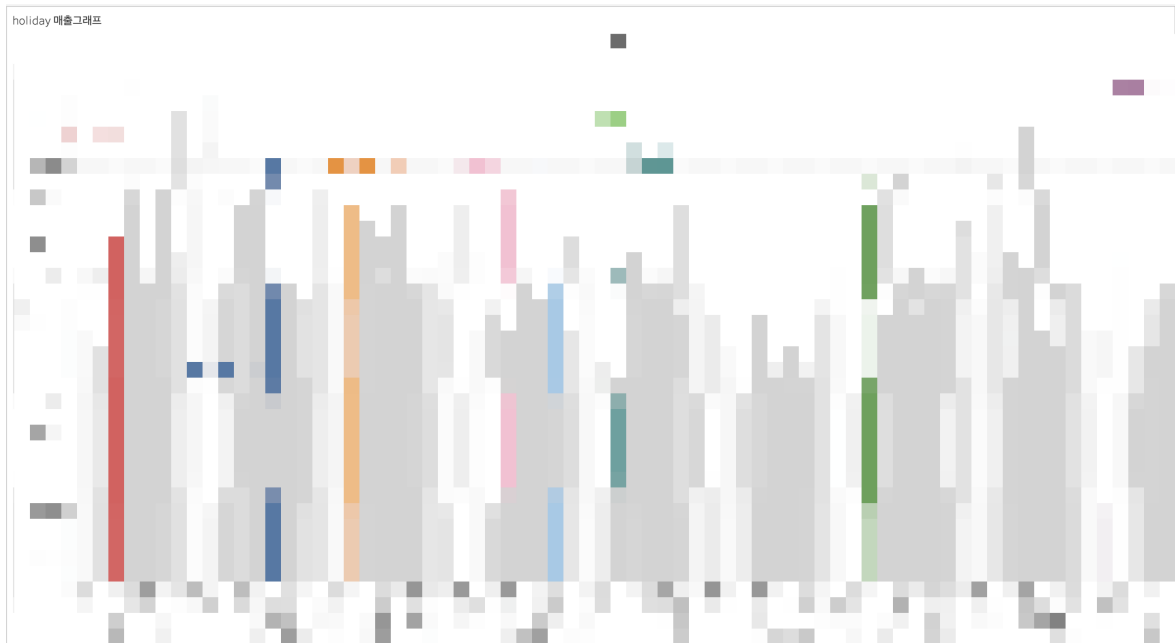


- 시간별 매출 합계

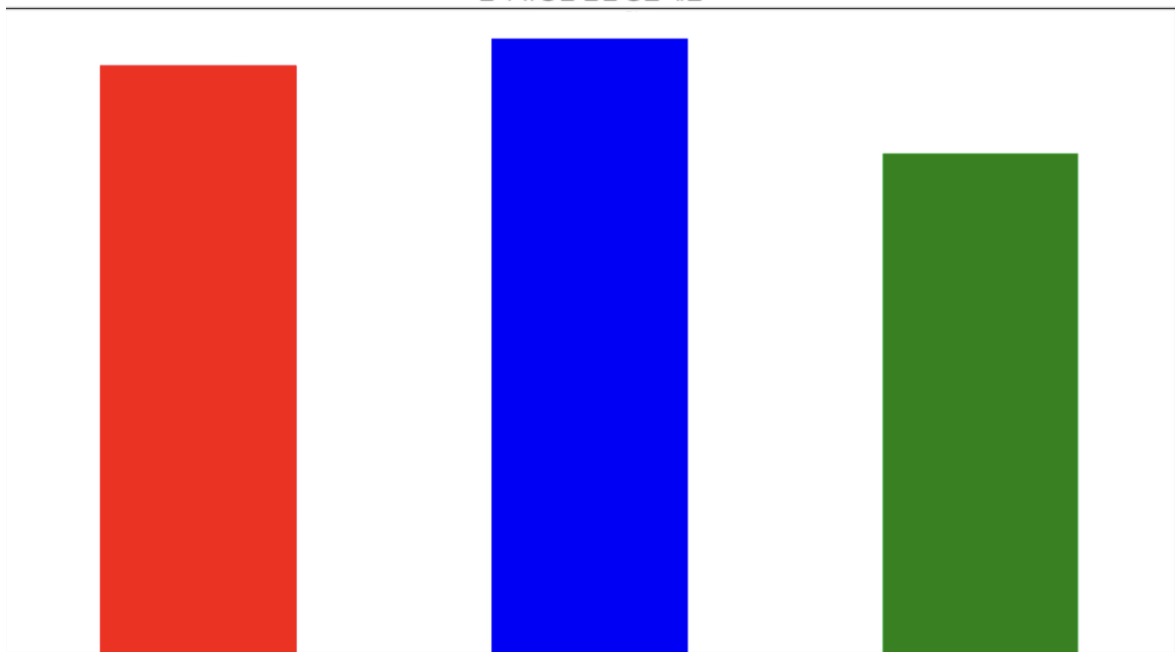


→ 커피 클래스를 운영 할 시기, 시간, 요일을 파악 하는데 도움을 주는 그래프들

- 공휴일 매출



일자 유형별 일별 평균 매출



→ 공휴일, 주말, 평일을 나누어 공휴일이 매출에 미치는 영향을 파악 할 수 있는 그래프
 그래프를 보아 공휴일은 매출에 큰 영향을 미치지 않는 것으로 판단

- 일매출과 기온



→ 기온에 따른 매출 비교 확인

기온이 너무 높거나 낮은 여름과 겨울에는 매출이 떨어지는 것으로 파악

🎯 솔루션(액션) 제안

‘김사홍’ 국가대표 바리스타와 함께하는 커피클래스 ☕

- 수강인원 : 8명
- 판매가격 : 1인 60,000 ~ 70,000원 (변경 가능)
- 커피 메뉴 설정
 - 여름: 아이스 시그니처 라떼
 - 겨울: 시그니처 카푸치노
- 운영 시기
 - 여름시즌 6, 7, 8월(세 달간) 진행
 - 겨울시즌 12, 1, 2월(세 달간) 진행
- 운영 주기 : 일주일에 하루, 금요일
- 운영 시간 : 영업 시간 종료 후 오후 7시 ~ 9시 (2시간)

[가격]

💰 1인 60,000 ~ 70,000원 (변경 가능)

외부 사례조사

- 서울, 경기, 제주 지역에서 각각 5곳의 커피 클래스 가격을 조사
- 평균 금액 서울 46,600원(1~3시간), 경기 54,000원(1.5~2시간), 제주 57,800원(2~3시간)

▼ 서울 - 46,600원 (평균)

와이드커피(머신커피) -> 40000원 / 1시간 / 1대1 or 단체

[https://smartstore.naver.com/_blending/products/5038573387?](https://smartstore.naver.com/_blending/products/5038573387?NaPm=ct%3Dlkw4ists%7Cci%3Dd71263c6b6274e681356b389fc0f9214959dde7b%7Ctr%3DsIsl%7Csn%3D1807767%7Ch)

[NaPm=ct%3Dlkw4ists%7Cci%3Dd71263c6b6274e681356b389fc0f9214959dde7b%7Ctr%3DsIsl%7Csn%3D1807767%7Ch](https://smartstore.naver.com/_blending/products/5038573387?NaPm=ct%3Dlkw4ists%7Cci%3Dd71263c6b6274e681356b389fc0f9214959dde7b%7Ctr%3DsIsl%7Csn%3D1807767%7Ch)

와디은커피(라떼아트) -> 88000원 / 1.5시간 / 1~10명

[https://smartstore.naver.com/_blending/products/5131964195?](https://smartstore.naver.com/_blending/products/5131964195?NaPm=ct%3Dlkw4rql4%7Cci%3D608846e2357d34a633230c142381bf30211d95c5%7Ctr%3DsIsl%7Csn%3D1807767%7C)

[NaPm=ct%3Dlkw4rql4%7Cci%3D608846e2357d34a633230c142381bf30211d95c5%7Ctr%3DsIsl%7Csn%3D1807767%7C](https://smartstore.naver.com/_blending/products/5131964195?NaPm=ct%3Dlkw4rql4%7Cci%3D608846e2357d34a633230c142381bf30211d95c5%7Ctr%3DsIsl%7Csn%3D1807767%7C)

커피앳웍스(브루잉) -> 10000원 / 1.5시간 / 8명

[https://smartstore.naver.com/cw_coffeeworks/products/7942827410?](https://smartstore.naver.com/cw_coffeeworks/products/7942827410?NaPm=ct%3Dlkw51j7k%7Cci%3Dce571651d6d15554d641926f3acfb1cd651204a8%7Ctr%3DsIsl%7Csn%3D910491%7Chl)
NaPm=ct%3Dlkw51j7k%7Cci%3Dce571651d6d15554d641926f3acfb1cd651204a8%7Ctr%3DsIsl%7Csn%3D910491%7Chl
OUR SARM -> 55000원 / 3시간 / 3~6명

[https://smartstore.naver.com/oursarm/products/5652033413?](https://smartstore.naver.com/oursarm/products/5652033413?NaPm=ct%3Dlkw522i0%7Cci%3Dda84d92943719a8dd933e937cd8fc37cfb08eb79%7Ctr%3DsIsl%7Csn%3D1081952%7Cl)
NaPm=ct%3Dlkw522i0%7Cci%3Dda84d92943719a8dd933e937cd8fc37cfb08eb79%7Ctr%3DsIsl%7Csn%3D1081952%7Cl
스트레도 로스터리(커피취향찾기) -> 40000원 / 2시간 / 2~6명

[https://smartstore.naver.com/caffe_stretto/products/4551485693?](https://smartstore.naver.com/caffe_stretto/products/4551485693?NaPm=ct%3Dlkw55a8o%7Cci%3D7d480127ba3334c84709b46862da31fac0ae95b4%7Ctr%3DsIsl%7Csn%3D965257%7C)
NaPm=ct%3Dlkw55a8o%7Cci%3D7d480127ba3334c84709b46862da31fac0ae95b4%7Ctr%3DsIsl%7Csn%3D965257%7C

▼ 경기 - 54,000원 (평균)

데프커피 -> 70000원 / 1.5~2시간 / 1:1 ~ 3:1

[https://smartstore.naver.com/sssd/products/6414729545?](https://smartstore.naver.com/sssd/products/6414729545?NaPm=ct%3Dlkw5chi0%7Cci%3D175204f3207649d5ad3bbcebb227d396b2a7a330%7Ctr%3DsIsl%7Csn%3D904529%7Cf)
NaPm=ct%3Dlkw5chi0%7Cci%3D175204f3207649d5ad3bbcebb227d396b2a7a330%7Ctr%3DsIsl%7Csn%3D904529%7Cf
데프커피 -> 60000원 / 1.5~2시간 / 1:1 ~ 3:1

[https://smartstore.naver.com/sssd/products/6382979844?](https://smartstore.naver.com/sssd/products/6382979844?NaPm=ct%3Dlkw5n14o%7Cci%3Ddab29210508861b1ead2d6b564ec32c5744f393d%7Ctr%3DsIsl%7Csn%3D904529%7C)
NaPm=ct%3Dlkw5n14o%7Cci%3Ddab29210508861b1ead2d6b564ec32c5744f393d%7Ctr%3DsIsl%7Csn%3D904529%7C
스테이 온 커피 -> 30000원 / 취미반: 1시간, 심화반 1.5시간 / 1~5명

[https://smartstore.naver.com/stayoncoffee/products/7876151136?](https://smartstore.naver.com/stayoncoffee/products/7876151136?NaPm=ct%3Dlkw5oeig%7Cci%3D7f4710684a292ab35966bd71fd0ed895c4794d1%7Ctr%3DsIsl%7Csn%3D7565674%7Cf)
NaPm=ct%3Dlkw5oeig%7Cci%3D7f4710684a292ab35966bd71fd0ed895c4794d1%7Ctr%3DsIsl%7Csn%3D7565674%7Cf
[경기광주/성남/용인]홈카페 입문 가정용 머신 기초 실습, 핸드드립 기초 원데이 클래스 -> 60000원 / 2시간 / 4명

[https://www.sssd.co.kr/m/class/detail/30972?](https://www.sssd.co.kr/m/class/detail/30972?utm_source=naver_shop&utm_medium=cpc&utm_campaign=cooking&utm_content=30972&NaPm=ct%3Dlkw5wqo0%7Cci)
utm_source=naver_shop&utm_medium=cpc&utm_campaign=cooking&utm_content=30972&NaPm=ct%3Dlkw5wqo0%7Cci
(주)커피사이언스 -> 50000원 / 3시간 / 2~6명명

[https://smartstore.naver.com/dayouforyou/products/8216976105?](https://smartstore.naver.com/dayouforyou/products/8216976105?NaPm=ct%3Dlkw6031c%7Cci%3D8941fba46d92a88d6b30f84193bead0916aa7b0e%7Ctr%3DsIsl%7Csn%3D903280%7Cf)
NaPm=ct%3Dlkw6031c%7Cci%3D8941fba46d92a88d6b30f84193bead0916aa7b0e%7Ctr%3DsIsl%7Csn%3D903280%7Cf

▼ 제주 - 57,800원 (평균)

커피배움-> 150000원(2~6명) / 2시간 / 2~6명

<https://www.sssd.co.kr/m/class/detail/34462>
크래커스 커피 나잇 -> 36000원 / 8명 / 2시간



[https://www.myrealtrip.com/offers/110143?](https://www.myrealtrip.com/offers/110143?utm_campaign=NshoppingTicket&utm_medium=CPC&utm_source=NaverShopping&NaPm=ct%3Dlkw53on4%7Cci%3D39a)
utm_campaign=NshoppingTicket&utm_medium=CPC&utm_source=NaverShopping&NaPm=ct%3Dlkw53on4%7Cci%3D39a
타오하우스 -> 100000원 / 1~3명 / 3시간

[https://smartstore.naver.com/nomadland/products/5019197304?](https://smartstore.naver.com/nomadland/products/5019197304?NaPm=ct%3Dlkw7md14%7Cci%3D90667b2c87a9ce35971dec0341f5a2fc829d13be%7Ctr%3DsIsl%7Csn%3D1029196%7C)
NaPm=ct%3Dlkw7md14%7Cci%3D90667b2c87a9ce35971dec0341f5a2fc829d13be%7Ctr%3DsIsl%7Csn%3D1029196%7C

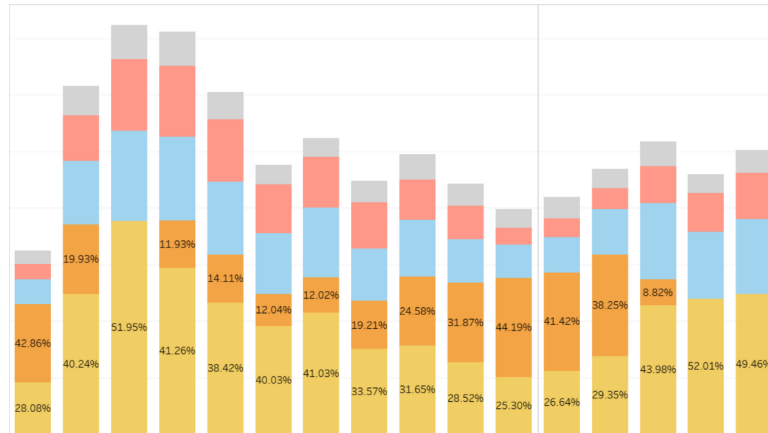
가격 제안

- 국가대표 바리스타가 최고급 원두를 사용해 진행한다는 프리미엄을 붙여서 6~7만원으로 책정

[커피 메뉴]

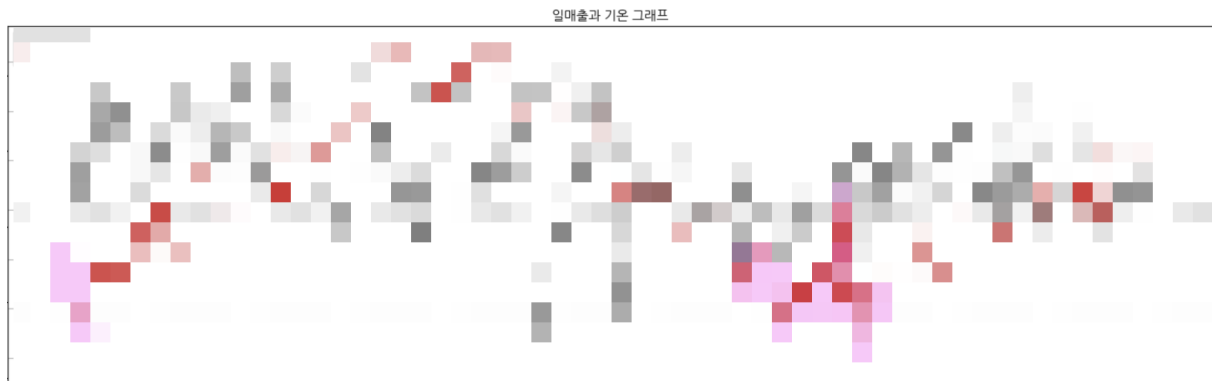
-  여름 - 아이스 시그니처 라떼
-  겨울 - 시그니처 카푸치노

[상위 5개 메뉴 월별 추이]



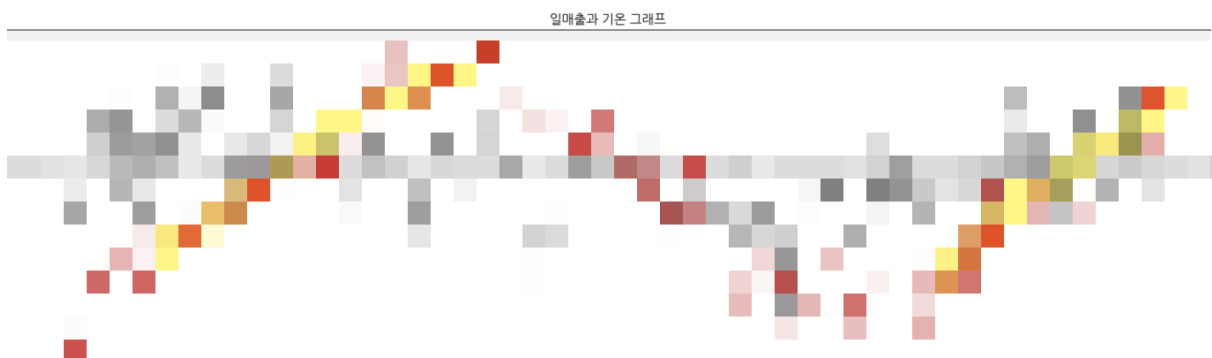
- 22년 4월에 시그니처 카푸치노 매출 없음(시그니처 카푸치노_디카페인/공연 시그니처 카푸치노만 있음)
- 23년 4~5월도 시그니처 카푸치노 매출 없음

[겨울] 메뉴 순위 분석



- '시그니처 카푸치노'의 매출이 1등인 시기 = 2월(22년), 11월, 12월, 1월, 2월(23년)
→ 월 평균 기온 10도 이하로 떨어진 달

[여름] 메뉴 순위 분석



아이스 시그니처 라떼

- '아이스 시그니처 라떼의 매출'이 1등인 시기: 22년 4월~10월 / 23년 3월~ 5월
- '아이스 시그니처 라떼'의 매출이 2등인 시기 : 22년 11월 12월 , 23년 1월 2월

= '시그니 카푸치노'의 매출이 2등인 시기 : 22년 11월 12월 , 23년 1월 2월

→ 아이스 시그니처 라떼 2등, 시그니처 카푸치노 1등의 시기가 정확히 겹침


아이스 시그니처 아메리카노


- 시그니처 커피라 일반 아이스 아메리카노보다는 매출이 높지만 1등을 한번도 기록한 적은 없음
- 그러나 3등 안에는 꼭 들어가는 스테디셀러 메뉴임
- 2등인 달 : 4월 5월 6월 7월 8월 9월 / 3월(23년) 4월(23년) 5월(23년)
- 3등인 달 : 2월(22년), 10월 11월 12월 1월 2월

아이스 아메리카노 클래식

- 22년 4월~9월, 23년 3월~5월 기간 동안 아이스 시그니처 아메리카노 2등, 아이스 아메리카노 클래식 3등이 겹침
- 날이 더워짐에 따라 아이스 아메리카노를 찾는 손님이 증가하지만 '아이스 시그니처 아메리카노'를 뛰어 넘은 적은 없음

[운영시기]

 여름시즌 6, 7, 8월(세 달간) 진행

 겨울시즌 12, 1, 2월(세 달간) 진행

[월별 커피템플 매출과 관광객 수]

매출이 떨어지는 추세가 확연 하게 나타나서 - 입도하는 관광객 수는 표로 넣어주시고, 이미지는 비공개 처리를 하겠습니다.

▼

날짜	외국인	내국인	총계	전년	증감율(%)
2022-02-01	3,148	1,026,355	1,029,503	793,768	29.7
2022-03-01	3,258	869,828	873,086	893,326	-2.3
2022-04-01	3,687	1,174,769	1,178,456	1,082,861	8.8
2022-05-01	4,574	1,301,963	1,306,537	1,136,452	15.0
2022-06-01	5,622	1,277,848	1,283,470	1,138,867	12.7
2022-07-01	6,487	1,256,845	1,263,332	1,131,512	11.6
2022-08-01	7,456	1,274,152	1,281,608	974,194	31.6
2022-09-01	7,658	1,051,499	1,059,157	872,396	21.4
2022-10-01	10,111	1,271,533	1,281,644	1,222,094	4.9
2022-11-01	14,529	1,135,039	1,149,568	1,204,344	-4.5
2022-12-01	16,013	996,326	1,012,339	1,090,607	-7.2
2023-01-01	15,849	1,016,716	1,032,565	1,170,802	-11.8
2023-02-01	12,929	1,041,370	1,054,299	1,029,503	2.4
2023-03-01	29,831	1,039,783	1,069,614	873,086	22.5
2023-04-01	41,606	1,115,662	1,157,268	1,178,456	-1.8
2023-05-01	49,693	1,131,015	1,180,708	1,306,537	-9.6

[22년 6월~10월]

- 관광객은 *월간 120만명 이상으로 꾸준히 제주도에 여행오는 것을 알 수 있음
 - 커피템플의 매출은 관광객 입도 수 대비 감소하는 것을 볼 수 있음
- 클래스 운영을 통해 떨어진 매출을 회복할 수 있고 좋은 원두와 경험을 제공함으로써 커피템플만의 가치를 높일 수 있음

(*9월은 '11호 태풍 힌남노'로 인해 관광객이 급감함 → 이상치 처리하였음)

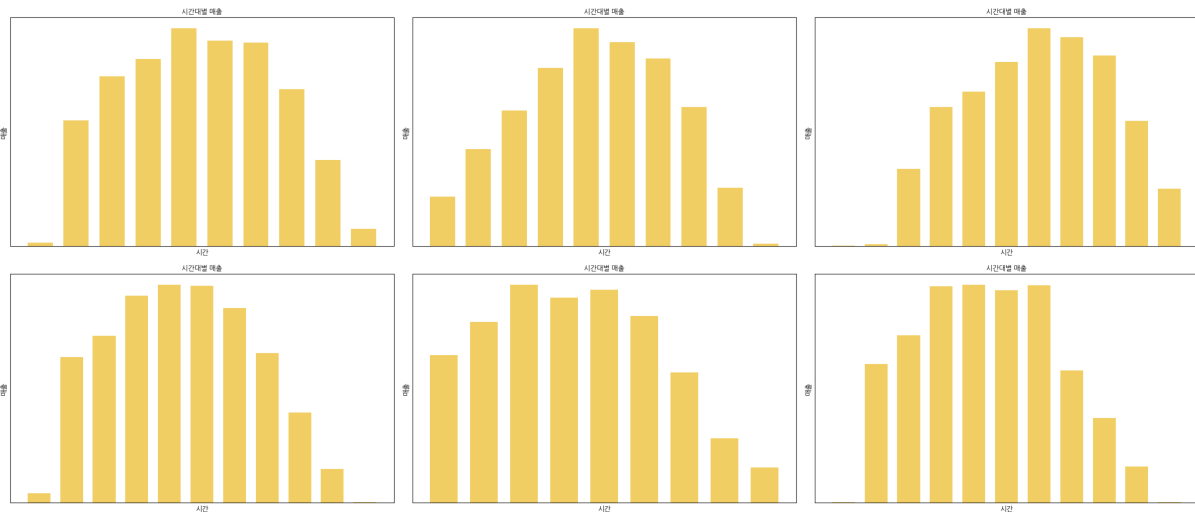
[22년 11월~23년 2월]

- 관광객의 수가 10월을 기점으로 겨울 내에 급격하게 줄어들고 마찬가지로 커피템플의 매출이 줄어들고 있음
→ 클래스를 제공하여 다른 카페들과 차별화 전략이 가능하기 때문에 제주도의 여러 카페 중 커피템플을 꼭 방문해야하는 이유를 제공함

[운영시간]

| ☺ 영업시간 이후 저녁 7시~9시

[여름, 겨울 시즌 시간대별 매출]



제외한 시간대

- 점심 시간 전후 (오후 12시~3시) : 일반 손님이 방문이 많고 매출이 가장 많기 때문에 제외함
- 오후 5시 : 5시 30분 이후로 주문이 불가하기 때문에 30분의 결측치가 발생하여 제외함
- 오전 10시, 오후 4시 : 아래와 같은 계산식을 통해 매출이 적은 시간대 매우 바쁠 것으로 추정하여 제외함
 - 6, 7, 8월의 오전 10시 평균 매출 n원, 일일 평균 매출은 약 n원임
 - 이를 시그니처 카푸치노(n원) 기준으로 계산하면 한 시간에 n잔~n잔 팔아야함

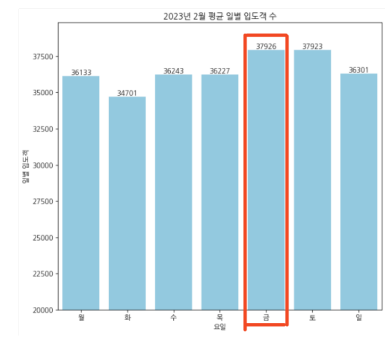
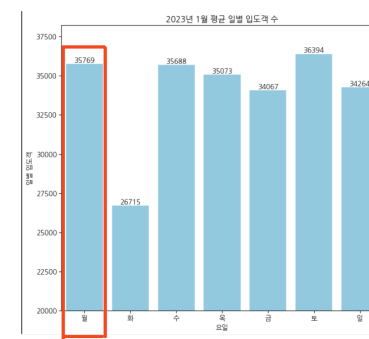
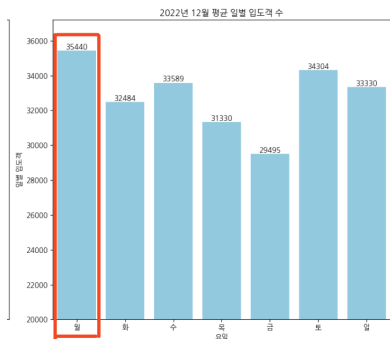
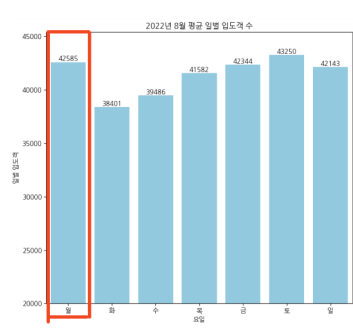
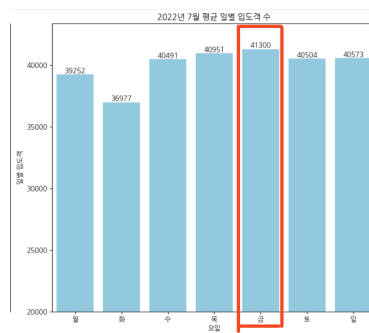
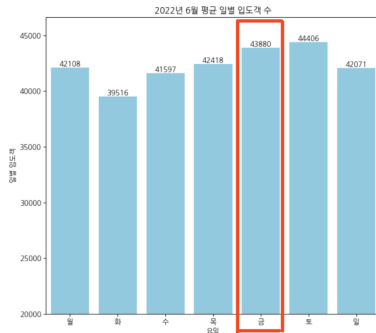
추천 시간대 : 영업시간 이후 저녁 7시~9시

- 영업시간에 클래스를 진행할 경우 카페방문 손님들이 매우 많아 소음이 큰 상황 또한 직원들의 리소스 부족으로 클래스 운영에 차질이 생길 수 있음
- 커피의 n배나 되는 가격을 지불하고 온 클래스 손님들의 몰입도와 만족도를 둘 다 높이기 위해 조용하고 프라이빗한 공간에서 진행해야 함

[운영주기]

| 17 일주일에 한 번, 매주 금요일 진행

[여름, 겨울 시즌 월별 평균 입도객 수]



[여름시즌]

- 22년 6월 - 금요일
- 22년 7월 - 금요일
- 22년 8월 - 월요일

[겨울시즌]

- 22년 12월 - 월요일
- 23년 1월 - 월요일
- 23년 2월 - 금요일

[분석결과 및 도출]

- 주말은 매출이 가장 높은 날이기 때문에 제외
- 화요일 제외 : 정기휴무일과 클래스 운영날이 같으면 방문객에 혼동을 야기할 수 있음
- 제주도 지리적 특성 상 월요일과 금요일을 주말에 붙여서 여행을 오는 사람들이 많음
→ 월요일 저녁에 비행기를 탄다면 클래스 참여에 어렵기 때문에 금요일로 설정

[일별 평균 매출] - 공휴일, 주말, 평일 비교

▼ 22년 2월~ 23년 5월 기간 공휴일 파악

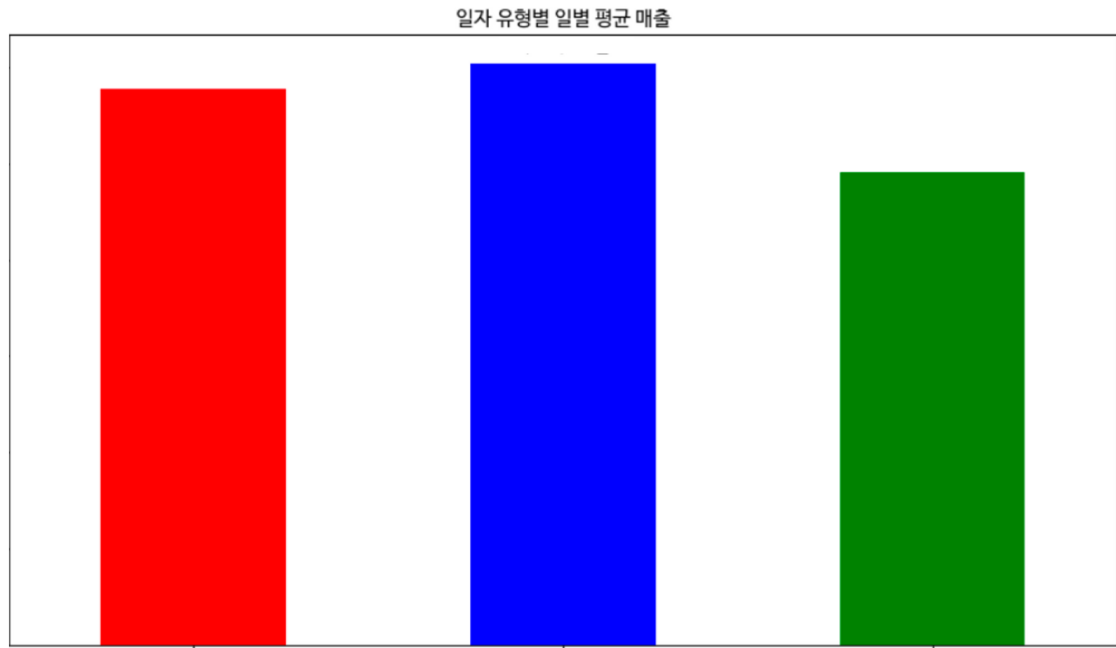
3일 이상 연휴(징검다리 연휴 포함)

2022년

- 설날 (1월 29일 30일 31일 - 2월 1일 2일)
- 삼일절 (2월 26일, 27일, 28일 ~ 3월 1일)
- 대통령선거 (3월 9일/수요일)
- 어린이날 (5월 5일, 6일, 7일, 8일)
- 지방선거 (6월 1일/수요일)
- 현충일(6월 4일, 5일, 6일)
- 광복절 (8월 13일, 14일, 15일)
- 추석 (9월 9일, 10일, 11일, 12일)
- 개천절 (10월 1일, 2일, 3일)
- 한글날 (10월 8일, 9일, 10일)

2023년

- 설날 (1월 21일, 22일, 23일, 24일)
- 삼일절 (3월 1일/수요일)
- 어린이날 (5월 5일, 6일, 7일)
- 부처님오신날 (5월 27일, 28일, 29일)



- 공휴일의 일일 평균 매출(n만원) < 공휴일이 아닌 주말의 매출(n만원)
→ 공휴일은 매출에 큰 영향을 미치지 않는 것으로 판단함

3-7. 모델링 결과

- 선택한 모델에 대한 설명과 평가 결과를 작성합니다.

[\[DAS\]FP_매출예측_Prophet_0802.ipynb](#)

모델 : Prophet

시계열 그래프는 단기예측만 쓰고 X축의 명확한 날짜 - 시기는 비공개처리(삭제)하겠습니다.

코드

▼ 목적 매출 예측

```
!pip install koreanize-matplotlib
# 불필요한 경고 메시지 무시
import warnings
warnings.filterwarnings('ignore')

# 라이브러리 임포트
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```

# 한글 글꼴 설정
import koreanize_matplotlib

# 데이터프레임 출력용 함수
from IPython.display import display

# 숫자 출력 조정
# 넘파이 부동소수점 출력 자리수 설정
np.set_printoptions(suppress=True, precision=4)

# 판다스 부동소수점 출력 자리수 설정
pd.options.display.float_format = '{:.4f}'.format

# 데이터프레임 모든 필드 출력
pd.set_option("display.max_columns", None)

# 그래프 글꼴 크기 설정
plt.rcParams["font.size"] = 14

# 난수 시드
random_seed = 123

# prophet
from prophet import Prophet

df = pd.read_csv('/content/drive/SharedDrives/[DAS]FinalProject_5조/송유림/모델링_매출예측/[DAS]FP_일시별_일매출(기온, 계절, 주말).csv')

# '일시'와 '일매출' 필드만 추출해
# 필드명을 '일시'는 ds, '일매출'는 y로
# 바꾼 새 데이터프레임 df를 생성한다

# 데이터프레임 사본을 생성
df2 = df.copy()

# '일시'와 '일매출' 필드를 추출
df2 = df2[['일시', '일매출']]

# 필드명을 교체
df2.columns = ['ds', 'y']

# 데이터분할: 8:2

# 입력 데이터
x_train = df2[:422]
x_test = df2[422:]

# 날짜 데이터 분할(그래프 출력용)
dates_test = x_test['ds']

# 모델 결정
m1 = Prophet(yearly_seasonality=True, weekly_seasonality=True,
             daily_seasonality=False,
             seasonality_mode='additive')

# 학습
m1.fit(x_train)

# 예측용 데이터 생성
# (날짜 필드 ds만 들어있는 데이터프레임)
# periods: 예측을 원하는 기간의 일수
future1 = m1.make_future_dataframe(periods=31, freq='D')

# 결과 확인
display(future1.head())
display(future1.tail())

# 예측
# 결과는 데이터프레임이 반환된다
fcst1 = m1.predict(future1)

```

▼ 매출 예측 그래프

```

# 학습 데이터, 검증 데이터 전체를 그래프로 출력
fig, ax = plt.subplots(figsize=(10, 6))

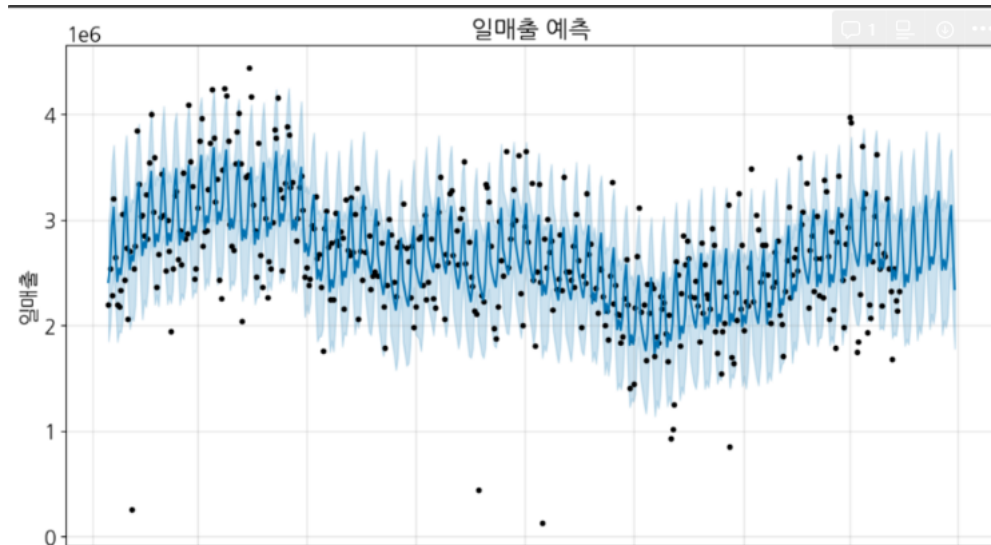
# 예측 결과를 그래프로 출력(프로phet 제공 함수)
m1.plot(fcst1, ax=ax)

# 제목 및 캡션 설정
ax.set_title('일매출 예측')

```

```
ax.set_xlabel('일시')
ax.set_ylabel('일매출')

# 그래프 출력
plt.show()
```



▼ 매출 예측 시계열 그래프

```
# 시계열 그래프 그리기
import matplotlib.dates as mdates
fig, ax = plt.subplots(figsize=(8, 4))

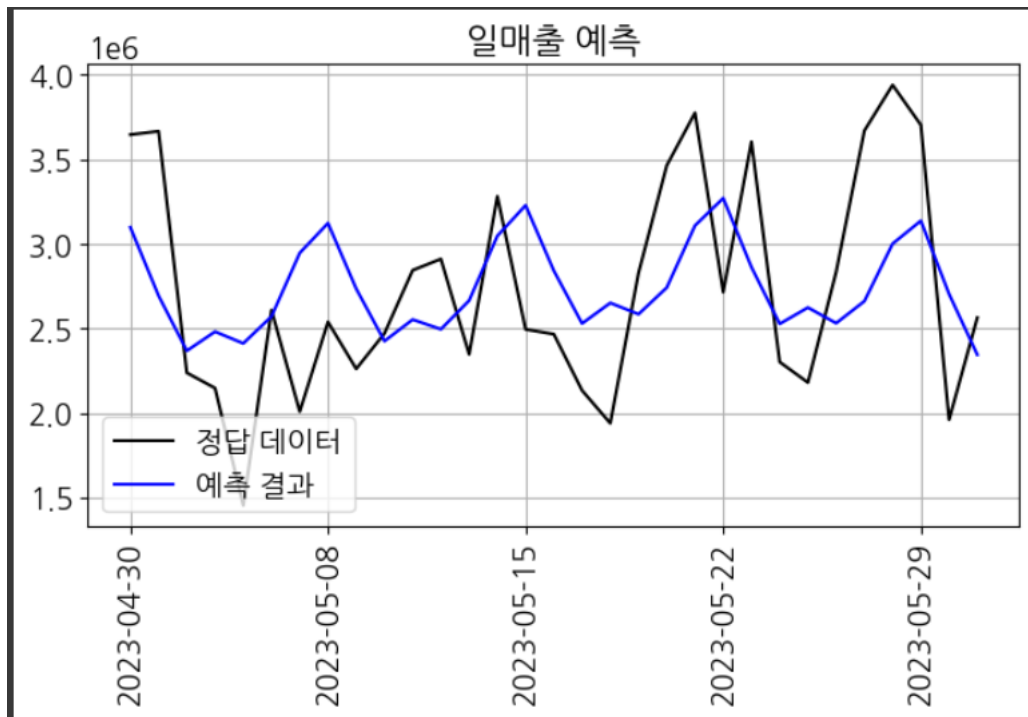
# 그래프 그리기
ax.plot(dates_test, ytest1, label='정답 데이터', c='k')
ax.plot(dates_test, ypred1, label='예측 결과', c='b')

# 날짜 눈금 표시
# 목요일마다 날짜를 출력한다
weeks = mdates.WeekdayLocator(byweekday=mdates.TH)
ax.xaxis.set_major_locator(weeks)

# 날짜 표기를 90도 기울임
ax.tick_params(axis='x', rotation=90)

# 그리드 그리기
ax.grid()
ax.legend()
ax.set_title('일매출 예측')

# 그래프 출력
plt.show()
```



▼ 튜닝(주말 반영)

```
# 주말, 금요일, 월요일에 해당하는 날짜 추출
df_holiday = df[df['주말'] == 1]
holidays = df_holiday['일시'].values

# 데이터프레임으로 변환
df_add = pd.DataFrame({'holiday': 'holi',
                       'ds': holidays,
                       'lower_window': 0,
                       'upper_window': 1
                      })

# 주말을 모델에 반영하기

# 알고리즘 선택
# holidays 파라미터를 추가해 새로운 모델 m2를 생성한다
m2 = Prophet(yearly_seasonality=True,
             weekly_seasonality=True, daily_seasonality=False,
             holidays = df_add, seasonality_mode='additive')

# 학습
m2 = m2.fit(x_train)

# 예측
fcst2 = m2.predict(future1)

# 요소별 그래프 그리기
fig = m2.plot_components(fcst2)
plt.show()
```

```
# R2값 계산하기

# fcst2에서 예측 부분만을 추출
ypred2 = fcst2[-31:]['yhat'].values

# R2값 계산
score2 = r2_score(ytest1, ypred2)

# 결과 확인
r2_text2 = f'R2 score:{score2:.4f}'
print(r2_text2)

# 시계열 그래프 그리기
import matplotlib.dates as mdates
fig, ax = plt.subplots(figsize=(8, 4))
```

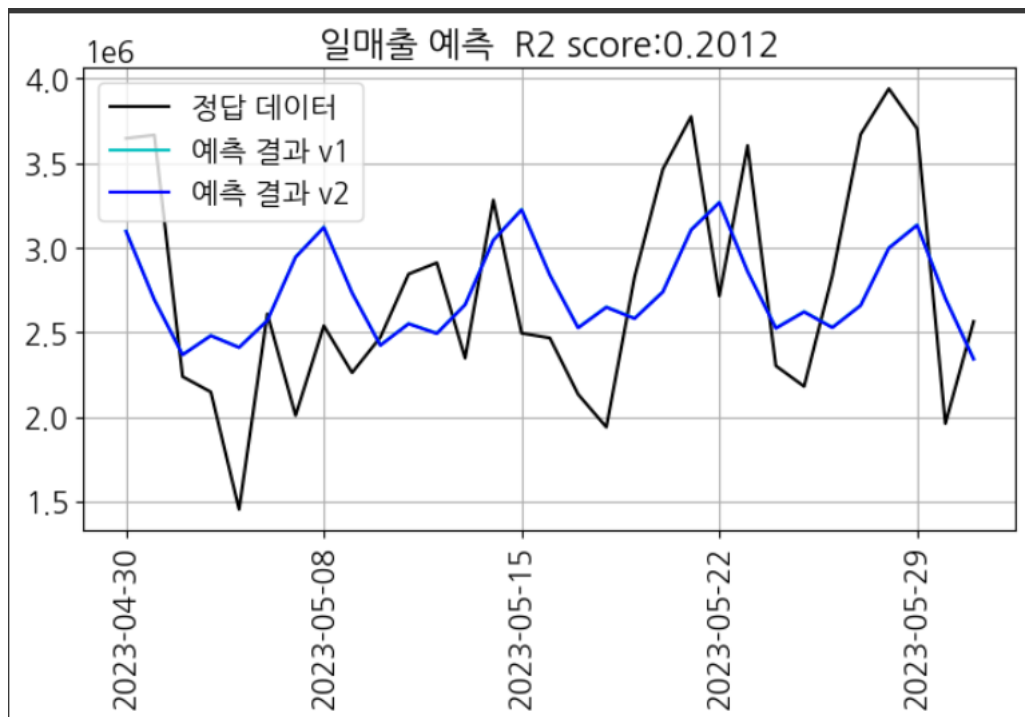
```
# 그래프 그리기
ax.plot(dates_test, ytest1, label='정답 데이터', c='k')
ax.plot(dates_test, ypred1, label='예측 결과 v1', c='c')
ax.plot(dates_test, ypred2, label='예측 결과 v2', c='b')

# 날짜 눈금 표시
# 목요일마다 날짜를 출력한다
weeks = mdates.WeekdayLocator(byweekday=mdates.TH)
ax.xaxis.set_major_locator(weeks)

# 날짜 표기를 90도 기울임
ax.tick_params(axis='x', rotation=90)

# 그리드 표시
ax.grid()
ax.legend()
ax.set_title('일매출 예측 ' + r2_text2)

# 화면에 출력
plt.show()
```



▼ 튜닝(기온 추가)

```
# 학습 데이터에 '기온' 추가하기
df3 = pd.concat([df2, df[['기온(°C)']], axis=1)

# 입력 데이터 분할
x2_train = df3[:422]
x2_test = df3[422:]

# 결과 확인
display(x2_train.tail())

# 알고리즘 선택
m3 = Prophet(yearly_seasonality=True,
              weekly_seasonality=True, daily_seasonality=False,
              seasonality_mode='additive', holidays = df_add)

# add_regressor 함수를 이용해 '기온(°C)'을 모델에 추가
m3.add_regressor('기온(°C)')

# 학습
m3.fit(x2_train)

# 예측용 입력 데이터 생성
future3 = df3[['ds', '기온(°C)']]
```



```
# 예측
fcst3 = m3.predict(future3)

# 요소별 그래프 그리기
fig = m3.plot_components(fcst3)
plt.show()
```

▼ 튜닝 시계열 그래프

```
# R2값 계산하기

# fcst에서 예측 부분만을 추출
ypred3 = fcst3[-31:][['yhat']].values
score3 = r2_score(ytest1, ypred3)

# 결과 확인
r2_text3 = f'R2 score:{score3:.4f}'
print(r2_text3)

# 시계열 그래프 그리기
import matplotlib.dates as mdates
fig, ax = plt.subplots(figsize=(8, 4))

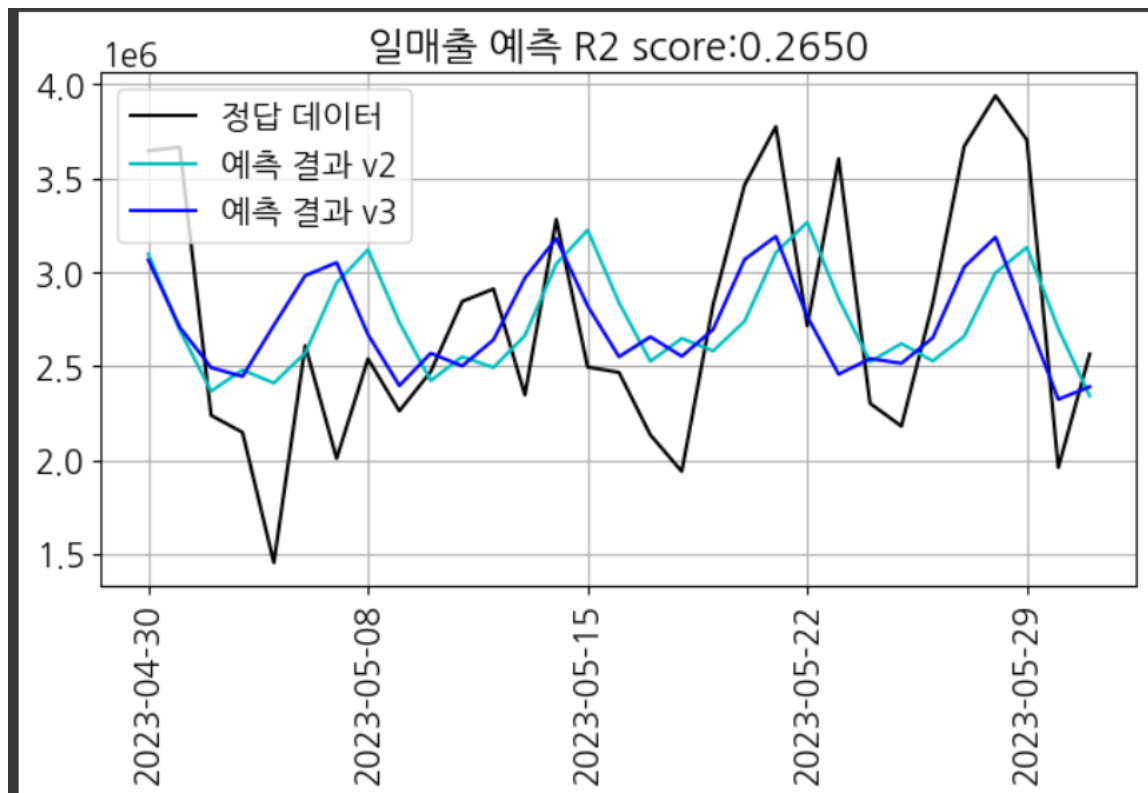
# 그래프 그리기
ax.plot(dates_test, ytest1, label='정답 데이터', c='k')
ax.plot(dates_test, ypred2, label='예측 결과 v2', c='c')
ax.plot(dates_test, ypred3, label='예측 결과 v3', c='b')

# 날짜 눈금 표시
# 목요일마다 날짜를 출력한다
weeks = mdates.WeekdayLocator(byweekday=mdates.TH)
ax.xaxis.set_major_locator(weeks)

# 날짜 표기를 90도 기울임
ax.tick_params(axis='x', rotation=90)

# 그리드 표시
ax.grid()
ax.legend()
ax.set_title('일매출 예측 ' + r2_text3)

# 화면에 출력
plt.show()
```



```

# Applying Prophet Model

m4 = Prophet(
    # 1) linear, nonlinear
    growth='linear',

    # 2) Trend
    changepoints=None, # CP가 발생하는 시점들의 list ['2012-01-01']
    n_changepoints=25, # CP의 수
    changepoint_range=0.8, # CP의 기존 데이터 수 대비 최대 비율
    changepoint_prior_scale=0.05, # CP 추정 민감도로 높을수록 민감

    # 3) Seasonality
    seasonality_mode='additive', # 계절성 모델: 'additive' or 'multiplicative'
    seasonality_prior_scale=10, # 계절성 추정 민감도로 높을수록 민감
    yearly_seasonality= True, # 연계절성
    weekly_seasonality= True, # 월계절성
    daily_seasonality= False, # 일계절성

    # 4) Holiday
    holidays=df_add, # 휴일 또는 이벤트 시점
    holidays_prior_scale= 10, # 휴일 추정 민감도로 높을수록 민감

    # 5) Others
    interval_width=0.4, # 추세 예측 정확도 구간범위
    mcmc_samples=10 # 계절성 예측 정확도 제어
)

# add_regressor 함수를 이용해 '기온(°C)'을 모델에 추가
m4.add_regressor('기온(°C)')

# 학습
m4.fit(x2_train)

# 예측용 입력 데이터 생성
future4 = df3[['ds', '기온(°C)']]

# 예측
fcst4 = m4.predict(future4)

# R2값 계산하기

# fcst에서 예측 부분만을 추출
ypred4 = fcst4[-31:][['yhat']].values
score4 = r2_score(ytest1, ypred4)

# 결과 확인
r2_text4 = f'R2 score:{score4:.4f}'
print(r2_text4)

# 시계열 그래프 그리기
import matplotlib.dates as mdates
fig, ax = plt.subplots(figsize=(8, 4))

# 그래프 그리기
ax.plot(dates_test, ytest1, label='정답 데이터', c='k')
ax.plot(dates_test, ypred4, label='예측 결과 v4', c='c')

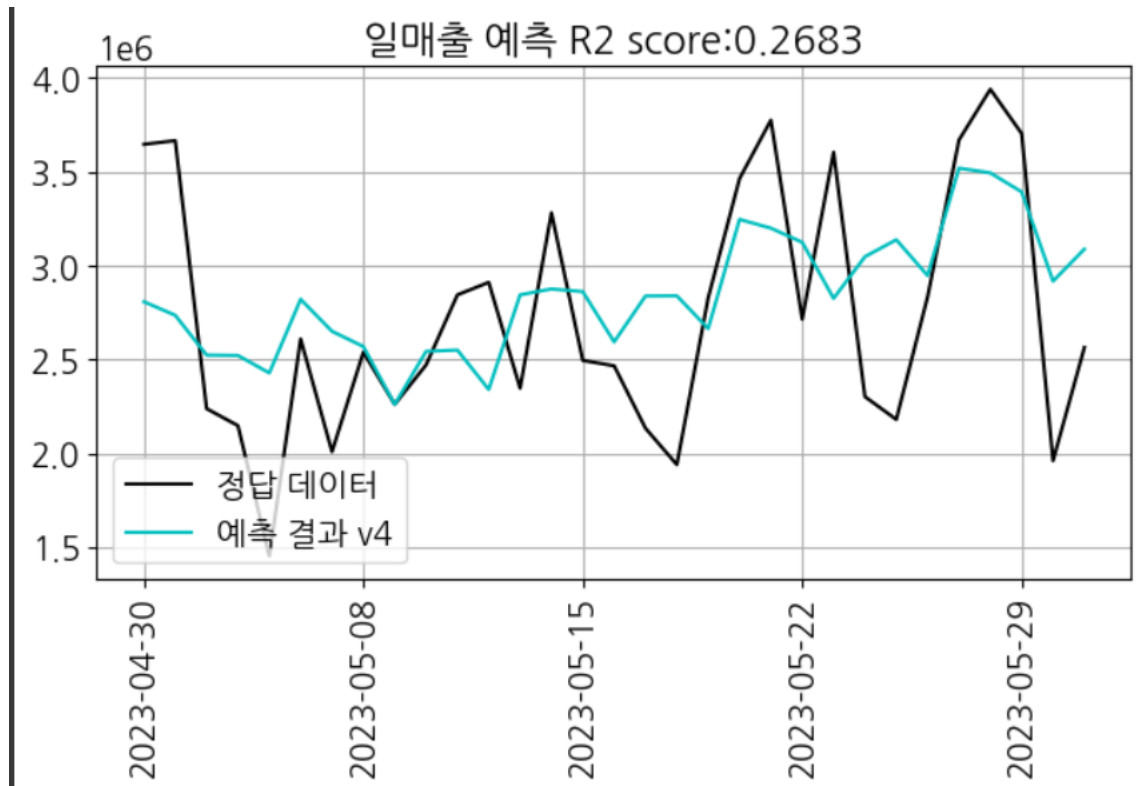
# 날짜 눈금 표시
# 목요일마다 날짜를 출력한다
weeks = mdates.WeekdayLocator(byweekday=mdates.TH)
ax.xaxis.set_major_locator(weeks)

# 날짜 표기를 90도 기울임
ax.tick_params(axis='x', rotation=90)

# 그리드 표시
ax.grid()
ax.legend()
ax.set_title('일매출 예측 ' + r2_text4)

# 화면에 출력
plt.show()

```



결과

1. 처음 값 R2 score : 0.2013
2. 주말 반영 한 값 R2 score : 0.2012
3. 기온 반영 한 값 R2 score : 0.2650
4. # Applying Prophet Model 수정 후 R2 score : 0.2683

3-8. 본 프로젝트의 활용 방안 제시

[예상효과 및 결론]

[예상효과]

1. 높은 퀄리티의 원두와 풍부한 향의 커피를 체험해볼 수 있는 프리미엄 경험을 제공함으로써 진정성과 가치를 모두 담은 이미지를 구축
2. '국가대표 바리스타'의 이미지를 클래스와 연상시켜 '커피템플'의 브랜드를 높일 수 있고 이것은 제주도의 랜드마크 카페로 가는 첫 걸음이 될 것
3. 제주도의 입도객이 적은, 커피템플의 매출이 떨어지는 시기에 매출 상승을 견인할 수 있는 하나의 선택지

[결론]



제주도에 2호점을 낸다는 가정하에 실제로 활용가능한 솔루션을 제안해보았음
구체적인 운영방법, 커리큘럼은 변동이 있을 수 있겠지만 내부 매출과 외부 변수를 연관지어 분석에 도움을 드리고자 함

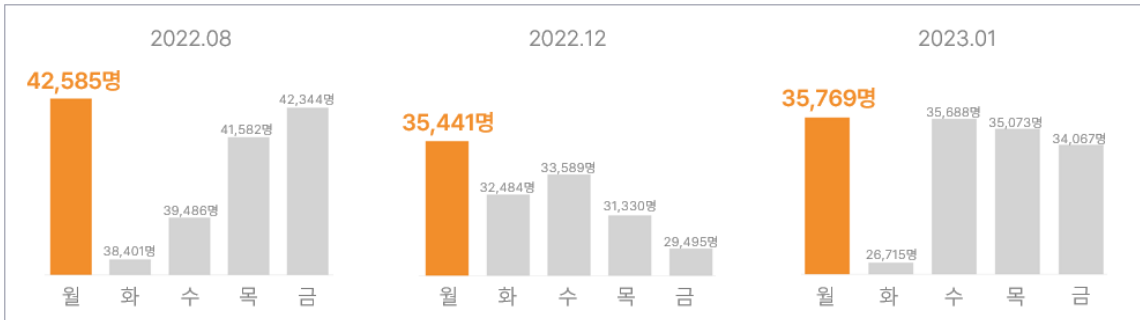
4. 프로젝트 회고 및 개선점

4-1. 현업자 피드백 + 각종 질문

발표에 대한 피드백

- 그래프를 직접 만들었을 때
: y축 수치를 생각할 때, 물결 표시를 하지 않으면 공격을 받을 수 있음

요일별 제주 입도 관광객 수 그래프



▼ 참고 자료

```
# 깨진 축

import matplotlib.pyplot as plt
import numpy as np

# [0, 0.2] 사이에 np.random.rand(30)*.2 사용하여 만든 30개 데이터
pts = np.array([
    0.015, 0.166, 0.133, 0.159, 0.041, 0.024, 0.195, 0.039, 0.161, 0.018,
    0.143, 0.056, 0.125, 0.096, 0.094, 0.051, 0.043, 0.021, 0.138, 0.075,
    0.109, 0.195, 0.050, 0.074, 0.079, 0.155, 0.020, 0.010, 0.061, 0.008])

pts[[3, 14]] += .8

# 우리가 단순히 pts를 구성한다면, 우리는 이상치 때문에 대부분의 세부사항들을 잃게 됨.
# 따라서 Y 축을 두 부분으로 'break'하거나 'cut-out'함
# 이상치에는 상단(ax)을, 대다수의 데이터에는 하단(ax2)을 사용
f, (ax, ax2) = plt.subplots(2, 1, sharex=True)

# 두 축을 동일한 데이터로 plot
ax.plot(pts)
ax2.plot(pts)

ax.set_ylim(.78, 1.) # 이상치만
ax2.set_ylim(0, .22) # 대부분의 데이터

# ax와 ax2 사이에 spine 숨기기
ax.spines['bottom'].set_visible(False)
ax2.spines['top'].set_visible(False)
ax.xaxis.tick_top()
ax.tick_params(labeltop=False) # 맨 위에 눈금 label 표시 안 함
ax2.xaxis.tick_bottom()

d = .015 # 축 좌표에서 사선을 만드는 방법
kwargs = dict(transform=ax.transAxes, color='k', clip_on=False)
ax.plot((-d, +d), (-d, +d), **kwargs) # 왼쪽 상단 사선
ax.plot((1 - d, 1 + d), (-d, +d), **kwargs) # 오른쪽 상단 사선

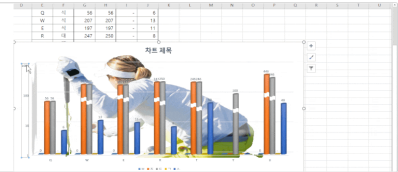
kwargs.update(transform=ax2.transAxes) # 밑의 축으로 전환
ax2.plot((-d, +d), (1 - d, 1 + d), **kwargs) # 왼쪽 하단 사선
ax2.plot((1 - d, 1 + d), (1 - d, 1 + d), **kwargs) # 오른쪽 하단 사선

plt.show()
```

엑셀 차트에 물결을 이용한 중간 생략 효과 주는 막대그래프와 사진 배경 삽입

오늘은 업무를 하다 보니, 목표와 실적을 표시하는 막대 차트를 만들게 되었는데 진행이 별로 되지 않아...

<https://m.blog.naver.com/romyok12/222592338694>



- 그래프 크기, 물결 표현 하나하나 넣을 때 그 효과를 생각하고 넣어야함 : 사소한 것도 신경 써서 괜한 책을 잡하지 않는 것이 중요함

4-2. 개선점 (팀 내에서 논의 및 합의된 개선 방향)

- 현재 1호점 데이터를 기반으로 분석, 제안한 솔루션이기 때문에 추후 2호점의 데이터와는 다를 수 있음
하지만 커피템플이라는 브랜드 가치 상승과 전 호점의 매출 관리 차원에서 활용될 수 있음
- 2호점이 개장한 후 실제 데이터를 가지고 분석을 진행하여 피드백을 통해 더 나은 방향과 솔루션을 도출하고자 함
(1호점 데이터를 기반으로 한 2호점 매출 예측이 얼마나 맞는지, 2호점은 어떻게 운영되고 있는지, 커피클래스를 운영한다면 유저 피드백을 통해 서비스를 고도화시킴)

5. 부록

5-1. 참고자료 및 출처

- 기상청, 기상자료개방포털 <https://data.kma.go.kr/data/grnd/selectAsosRltmList.do?pgmNo=36>
- 한국관광 데이터랩 datalab.visitkorea.or.kr
- 제주관광협회
<http://www.visitjeju.or.kr/web/bbs/bbsList.do;jsessionid=bYbxkvPGKZIXFFm3RhQlbiI7T1LVI7hAsngOi8B1pRn37cA1xPMBjh>
[bbsId=TOURSTAT](http://www.visitjeju.or.kr/web/bbs/bbsList.do;jsessionid=bYbxkvPGKZIXFFm3RhQlbiI7T1LVI7hAsngOi8B1pRn37cA1xPMBjh)
- 제주관광공사 https://ijto.or.kr/korean/pds_search/search.php?cid=184
- 관광지식정보시스템 <https://know.tour.go.kr/>
- 인천국제공항공사 <https://www.airport.kr/co/ko/cpr/statisticCategoryOfTimeSeries.do>