19장 실전 데이터 분석 프로젝트

■ 서울시에서 공개하는 업무추진비 데이터를 활용해 데이터 분석을 수행한다.

19.1 데이터 분석 프로세스

■ 1. 주제 선정
- 데이터 분석의 목표를 명확히 하고 이로부터 주제를 선정한다. 즉, 데이터 분석을 통해 얻고 싶은 결과가 무엇인지를 설정하는 것이다.

■ 2. 데이터 수집
- 주제가 선정되면 주제에 맞는 데이터를 수집해야 한다.
- 국내외 무료 공개 데이터 저장소
  - 공공 데이터 포텔(https://www.data.go.kr)
  - 서울시 정보소통광장(https://opengov.seoul.go.kr/)
  - 국가 통계 포털(http://kosis.kr/)
  - 세계 은행 오픈 데이터(https://data.worldbank.org/)
  - FRD(https://fred.stlouisfed.org/)
  - 캐글 사이트(https://www.kaggle.com)
  - U.S. Government's open data(https://www.data.gov/)

■ 3. 데이터 처리
- 데이터가 수집되면 다음은 데이터 분석이 편리하도록 데이터를 처리하는 과정을 거친다.
- 데이터 정제: 가공되지 않은 초기 데이터에서 부정확한 데이터를 찾아내고 수정하거나 제거하는 절차이다.
- 데이터 타이딩: 데이터 분석을 위해 데이터를 알아보기 쉽고 처리하기 편하도록 구조화하는 절차이다. 보통 데이터 정제 이후에 이뤄지는데, 이 과정을 거친 데이터를 깔끔한 데이터(Tidy data)라고 한다.

■ 4. 데이터 분석
- 앞 단계에서 처리된 데이터를 이용해 다양한 기법으로 데이터를 분석한다. 대표적인 데이터 분석 방법은 통계적 분석 방법이며 최근에는 머신러닝을 활용한 데이터 분석도 많이 수행한다.
- 데이터 분석 이후에는 분석 결과를 효과적으로 전달하기 위해 다양한 시각화 방법을 이용한다.

■ 5. 정보 도출
- 데이터 분석 과정으로 도출된 결과에서 의미를 발견하는 단계이다. 이 단계에서 앞의 모든 과정이 올바르게 이뤄졌는지 검증하는 과정이 필요하다.

19.2 데이터 획득, 처리, 시각화 심화

19.2.1 깃허브에서 파일 내려받기

■ requests를 이용해 깃허브에서 파일을 내려받고 저장하는 코드를 작성한다.

```
[ch19_project/ex01_github.py]
```

```
01  import requests
02
03  # 깃허브의 파일 URL
04  url = 'https://github.com/wikibook/python-for-data-analysis-rev/raw/master/readme.txt'
05
06  # URL에 해당하는 파일을 내려받음
07  r = requests.get(url)
08
09  # 파일을 저장할 폴더와 파일명을 지정
10  file_name = './ch19_project/readme.txt'
11
12  # 내려받은 파일을 지정한 폴더에 저장
13  with open(file_name, 'wb') as f:
14      f.write(r.content)
```

19.2.2 데이터에서 결측치 확인 및 처리

- 수집된 데이터에는 다양한 이유로 데이터 값이 누락될 수 있다. 이처럼 누락된 데이터를 결측치(Missing data)라고 하며, 데이터에 결측치가 있다면 데이터를 분석할 때 결과가 왜곡되거나 데이터 분석이 불가능할 수도 있다.따라서 데이터에서 어떠한 값이 결측치인지 아는 것이 중요하다.

- Pandas의 DataFrame 형식으로 가져온 데이터의 경우 isna() 혹은 isnull()을 통해 결측치 여부를 판단할 수 있다.
  - 18: isnull()을 이용해 결측치 여부를 확인한다.
  - 29: isnull()의 결과에 sum()을 수행하면 결측치 개수를 셀 수 있다.

[ch19_project/ex02_missing_data.py]

```
01  # 결측치 확인
02  import pandas as pd
03
04  data_file = "./ch19_project/missing_data_test.csv"
05
06  df = pd.read_csv(data_file, encoding="cp949", index_col="연도")
07
08  print(df)
09  '''
10          제품1  제품2    제품3    제품4
11  연도
12  2015  250.0  150    NaN    NaN
13  2016  200.0  160  170.0    NaN
14  2017  150.0  200  100.0  150.0
15  2018  120.0  230  130.0  170.0
16  2019    NaN  250  140.0    NaN
17  '''
18
19  print(df.isnull())
20  '''
21          제품1    제품2    제품3    제품4
22  연도
23  2015  False  False   True   True
24  2016  False  False  False   True
25  2017  False  False  False  False
26  2018  False  False  False  False
27  2019   True  False  False   True
28  '''
29
30  print(df.isnull().sum())
```

```
'''
제품1    1
제품2    0
제품3    1
제품4    3
dtype: int64
'''


# 결측치 처리
print(df.drop(index=[2019]))
'''
        제품1  제품2    제품3    제품4
연도
2015  250.0  150    NaN    NaN
2016  200.0  160  170.0    NaN
2017  150.0  200  100.0  150.0
2018  120.0  230  130.0  170.0
'''

print(df.drop(columns=['제품3', '제품4']))
'''
        제품1  제품2
연도
2015  250.0  150
2016  200.0  160
2017  150.0  200
2018  120.0  230
2019    NaN  250
'''

print(df.drop(index=[2018, 2019], columns=['제품3', '제품4']))
'''
        제품1  제품2
연도
2015  250.0  150
2016  200.0  160
2017  150.0  200
'''


print(df.dropna())  # df.dropna(axis=0)도 결과는 같습니다.
'''
        제품1  제품2    제품3    제품4
연도
2017  150.0  200  100.0  150.0
2018  120.0  230  130.0  170.0
'''

print(df.dropna(axis=0, subset=['제품1']))
'''
        제품1  제품2    제품3    제품4
연도
2015  250.0  150    NaN    NaN
2016  200.0  160  170.0    NaN
2017  150.0  200  100.0  150.0
2018  120.0  230  130.0  170.0
'''

print(df.dropna(axis=1))
'''
      제품2
연도
2015  150
2016  160
```

```
 96  2017  200
 97  2018  230
 98  2019  250
 99  '''
100
101  print(df.fillna(0))
102  '''
103          제품1  제품2    제품3    제품4
104  연도
105  2015  250.0  150    0.0    0.0
106  2016  200.0  160  170.0    0.0
107  2017  150.0  200  100.0  150.0
108  2018  120.0  230  130.0  170.0
109  2019    0.0  250  140.0    0.0
110  '''
111
112  print(df.fillna(method='bfill'))
113  '''
114          제품1  제품2    제품3    제품4
115  연도
116  2015  250.0  150  170.0  150.0
117  2016  200.0  160  170.0  150.0
118  2017  150.0  200  100.0  150.0
119  2018  120.0  230  130.0  170.0
120  2019    NaN  250  140.0    NaN
121  '''
122
123  print(df.fillna(method='ffill'))
124  '''
125          제품1  제품2    제품3    제품4
126  연도
127  2015  250.0  150    NaN    NaN
128  2016  200.0  160  170.0    NaN
129  2017  150.0  200  100.0  150.0
130  2018  120.0  230  130.0  170.0
131  2019  120.0  250  140.0  170.0
132  '''
133
134  values = {'제품1': 100,  '제품4': 400}
135  print(df.fillna(value=values))
136  '''
137          제품1  제품2    제품3    제품4
138  연도
139  2015  250.0  150    NaN  400.0
140  2016  200.0  160  170.0  400.0
141  2017  150.0  200  100.0  150.0
142  2018  120.0  230  130.0  170.0
143  2019  100.0  250  140.0  400.0
144  '''
```

19.2.3 데이터의 요약 및 재구성

■ CSV 데이터 파일(total_sales_data.csv)을 DataFrame 형식으로 읽어오는 코드는 다음과 같다.

[ch19_project/ex03_info.py]

```
01  import pandas as pd
02
03  data_file = "./ch19_project/total_sales_data.csv"
04
```

```
df_sales = pd.read_csv(data_file)
print(df_sales)
'''
  매장명  제품종류 모델명  판매  재고
0   A  스마트폰  S1   1   2
1   A  스마트폰  S2   2   5
2   A    TV  V1   3   5
3   B  스마트폰  S2   4   6
4   B  스마트폰  S1   5   8
5   B    TV  V1   6   9
6   C  스마트폰  S2   2   4
7   C    TV  V1   3   6
8   C    TV  V2   7   9
'''

print(df_sales.info())
'''
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   매장명     9 non-null      object
 1   제품종류    9 non-null      object
 2   모델명     9 non-null      object
 3   판매      9 non-null      int64
 4   재고      9 non-null      int64
dtypes: int64(2), object(3)
memory usage: 488.0+ bytes
None
'''

print(df_sales['매장명'].value_counts())
'''
C    3
B    3
A    3
Name: 매장명, dtype: int64
'''

print(df_sales['제품종류'].value_counts())
'''
스마트폰    5
TV      4
Name: 제품종류, dtype: int64
'''


print(df_sales.pivot_table(index=["매장명", "제품종류", "모델명"],
                     values =["판매","재고"], aggfunc='sum'))
'''
              재고  판매
매장명 제품종류 모델명
A   TV   V1    5   3
    스마트폰 S1    2   1
         S2    5   2
B   TV   V1    9   6
    스마트폰 S1    8   5
         S2    6   4
C   TV   V1    6   3
         V2    9   7
    스마트폰 S2    4   2
'''

print(df_sales.pivot_table(index=["매장명"], columns = ["제품종류"],
```

```
                    values =["판매","재고"], aggfunc='sum'))
'''
      재고       판매
제품종류  TV 스마트폰  TV 스마트폰
매장명
A      5    7   3    3
B      9   14   6    9
C     15    4  10    2
'''

print(df_sales.pivot_table(index=["매장명"], columns = ["제품종류"],
                    values =["판매","재고"], aggfunc='count'))
'''
     재고      판매
제품종류 TV 스마트폰 TV 스마트폰
매장명
A     1    2  1    2
B     1    2  1    2
C     2    1  2    1
'''
```

19.2.4 워드 클라우드를 이용한 데이터 시각화

■ 워드 클라우드는 텍스트 데이터에서 출현 빈도가 높은 단어는 크게 표시하고 출현 빈도가 낮은 단어는 작게 표시하는 방법으로 데이터를 시각화한다.

■ 워드 클라우드 설치

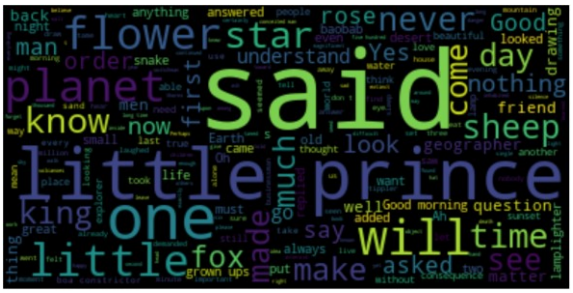
```
D:\dev\workspace\python>pip install wordcloud
Collecting wordcloud
  Downloading wordcloud-1.8.1-cp38-cp38-win_amd64.whl (155 kB)
     |████████████████████████████████████████████████████████████████| 155 kB 6.8 MB/s
…(생략)…
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.1
```

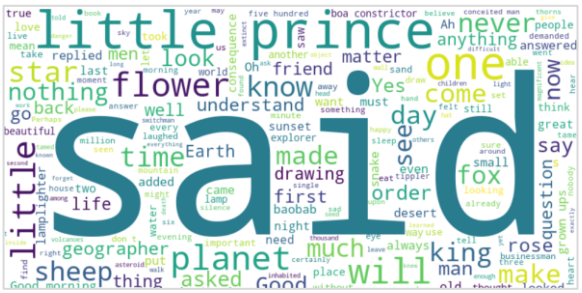
■ 다음은 텍스트 파일로 워드 클라우드 이미지를 생성해서 화면에 출력하는 코드이다.

[ch19_project/ex04_wordcloud.py]

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

file_name = './ch19_project/littleprince_djvu.txt'

with open(file_name) as f: # 파일을 읽기 모드로 열기
    text = f.read() # 파일의 내용 읽어오기

# 워드 클라우드의 이미지를 생성합니다.
wordcloud_image = WordCloud().generate(text)

# 생성한 워드 클라우드 이미지를 화면에 표시합니다.
plt.imshow(wordcloud_image, interpolation='bilinear')
plt.axis("off")
plt.show()


```

```
wordcloud_image = WordCloud(background_color='white', max_font_size=300, width=800,
height=400).generate(text)

plt.imshow(wordcloud_image, interpolation="bilinear")
plt.axis("off")
plt.show()
```





19.3 실전 데이터(서울시 업무추진비) 분석

- 서울시는 업무의 투명성을 위해 정보소통광장(http://opengov.seoul.go.kr/)을 통해 결재문서, 회의정보, 사전공표 등의 행정 정보를 공개하고 있다. 이 가운데 사전공표에 있는 업무추진비 데이터를 이용해 데이터 분석을 진행한다.

19.3.1 데이터 분석의 주제 선정

- 이번에 수행할 데이터 분석의 주제는 서울시에서 집행한 업무추진비 분석으로 선정한다.

- 서울시는 정보소통광장을 통해 시장단을 비롯해서 4급 부서장 이상 간부직 공무원의 업무추진비 집행 내역(https://opengov.seoul.go.kr/expense)을 공개하고 있다.

부서별 사용내역 | 기간별 이용내역

부서: 서울시본청 | 서울특별시장

범위: 2021년 | 10월 | 검색 | 2021년10월 다운로드 2016년 10월 이전자료는 기간별 이용내역에서 확인하세요

서울특별시장 기관운영, 시책추진 업무추진비 사용내역 엑셀 다운로드

연번	구분	부서명	사용일시	사용장소	사용목적	사용금액	사용자 및 인원	결제방법	비목
1		행정국 총무과(서울특별시장)	2021-10-01 13:15	서울시청 간담회장(서울 중구 세종대로 110)	시정현안 관련 의견수렴 간담회	150,000	시장 외 5명	카드	시책
2	현업·우수부서 격려 등	행정국 총무과(서울특별시장)	2021-10-01 16:02	피자헛(서울특별시 종로구 수표로 84)	현안 관련 격무부서 격려	362,800	자치행정과	제로페이	기관
3		행정국 총무과(서울특별시장)	2021-10-01 19:55	한양동, 꼬프(Cof1)(서울특별시 중구 정동길 21-15)	시정 추진 관련 자문 간담회	173,000	시장 외 5명	카드	시책

■ 다음과 같은 세부 주제를 선정해서 데이터 분석을 진행한다.

- 연도별 추이 분석
- 월별 집행금액 분석
- 부서별 집행 내역 분석
- 요일별 및 시간대별 집행 내역 분석

19.3.2 데이터 수집

■ 서울시 행정정보 공개 깃허브(https://github.com/seoul-opengov/opengov)에 있는 업무추진비 집행내역을 가져온다.

[ch19_project/ex05_데이터수집.py]

```
01  # 데이터 수집
02  import glob
03  import requests
04  import os
05  import pathlib
06
07  #인자: 확장자, 연도, 내려받을 폴더
08
09
10  def get_seoul_expense_list(extension, year, data_folder):
11
12      # 깃허브의 데이터 위치 지정
13      # ex) 'https://github.com/seoul-opengov/opengov/raw/master/expense_list2017/'
14      expense_list_year_url = 'https://github.com/seoul-opengov/opengov/raw/master/expense_list' + \
15          str(year) + '/'
16
17      # 데이터를 내려받을 폴더 지정
18      # ex) 'C:/myPyCode/data/seoul_expense/2017/'
19      expense_list_year_dir = data_folder + str(year) + '/'
20
21      # 내려받을 폴더가 없다면 폴더 생성
22      if(os.path.isdir(expense_list_year_dir)):
23          print("데이터 폴더({0})가 이미 있습니다. {0}년 데이터의 다운로드를 시작합니다.".format(year))
24      else:
25          print("데이터 폴더({0})가 없어서 생성했습니다. {0}년 데이터의 다운로드를 시작합니
26  다.".format(year))
27          # 폴더 생성
28          pathlib.Path(expense_list_year_dir).mkdir(parents=True, exist_ok=True)
29
30      # 지정한 폴더로 1월 ~ 12월 업무추진비 파일을 다운로드
31      for k in range(12):
32          file_name = '{0}{1:02d}_expense_list.{2}'.format(year, k+1, extension)
```

```
        url = expense_list_year_url + file_name
        print(url)
        r = requests.get(url)
        with open(expense_list_year_dir + file_name, 'wb') as f:
            f.write(r.content)


# 내려받을 업무추진비 데이터의 파일 형식을 지정
extension = "csv"

# 내려받을 업무추진비 데이터의 연도를 지정
year = 2017

# 내려받을 업무추진비 데이터의 폴더를 지정
data_folder = './ch19_project/seoul_expense/'

# 함수를 실행
get_seoul_expense_list(extension, year, data_folder)


path_name = './ch19_project/seoul_expense/2017/'  # 폴더 이름

# 지정 폴더에서 파일명에 list.csv가 포함된 파일만 지정
file_name_for_glob = path_name + "*list.csv"

csv_files = []
for csv_file in glob.glob(file_name_for_glob):
    # 반환값에서 폴더는 제거하고 파일 이름만 추출
    csv_files.append(csv_file.split("\\")[-1])

print("[폴더 이름]", path_name)  # 폴더명 출력
print("* CSV 파일:", csv_files)
'''
[폴더 이름] ./ch19_project/seoul_expense/2017/
*  CSV  파일:  ['201701_expense_list.csv',  '201702_expense_list.csv',  '201703_expense_list.csv',
'201704_expense_list.csv',            '201705_expense_list.csv',            '201706_expense_list.csv',
'201707_expense_list.csv',            '201708_expense_list.csv',            '201709_expense_list.csv',
'201710_expense_list.csv', '201711_expense_list.csv', '201712_expense_list.csv']
'''


data_folder = './ch19_project/seoul_expense/'

years = [2017, 2018, 2019]  # 다운로드받을 연도를 지정

extension = "csv"
# extension = "xlsx"
# extension = "xml"

for year in years:
    get_seoul_expense_list(extension, year, data_folder)

print("모든 데이터를 다운로드 받았습니다.")


data_folder = './ch19_project/seoul_expense/'

years = [2017, 2018, 2019]  # 다운로드받을 연도를 지정

for year in years:
    path_name = data_folder + str(year) + "/"  # 연도별 폴더명을 지정

    # 지정 폴더에서 파일명에 list.csv가 포함된 파일만 지정
    file_name_for_glob = path_name + "*list.csv"

```

```
    csv_files = []
    for csv_file in glob.glob(file_name_for_glob):
        # 반환값에서 폴더는 제거하고 파일명만 추출
        csv_files.append(csv_file.split("\\")[-1])

    print("[폴더 이름]", path_name)  # 폴더명 출력
    print("* CSV 파일:", csv_files)
'''
[폴더 이름] ./ch19_project/seoul_expense/2017/
*  CSV  파일:  ['201701_expense_list.csv',  '201702_expense_list.csv',  '201703_expense_list.csv',
'201704_expense_list.csv',            '201705_expense_list.csv',            '201706_expense_list.csv',
'201707_expense_list.csv',            '201708_expense_list.csv',            '201709_expense_list.csv',
'201710_expense_list.csv', '201711_expense_list.csv', '201712_expense_list.csv']
[폴더 이름] ./ch19_project/seoul_expense/2018/
*  CSV  파일:  ['201801_expense_list.csv',  '201802_expense_list.csv',  '201803_expense_list.csv',
'201804_expense_list.csv',            '201805_expense_list.csv',            '201806_expense_list.csv',
'201807_expense_list.csv',            '201808_expense_list.csv',            '201809_expense_list.csv',
'201810_expense_list.csv', '201811_expense_list.csv', '201812_expense_list.csv']
[폴더 이름] ./ch19_project/seoul_expense/2019/
*  CSV  파일:  ['201901_expense_list.csv',  '201902_expense_list.csv',  '201903_expense_list.csv',
'201904_expense_list.csv',            '201905_expense_list.csv',            '201906_expense_list.csv',
'201907_expense_list.csv',            '201908_expense_list.csv',            '201909_expense_list.csv',
'201910_expense_list.csv', '201911_expense_list.csv', '201912_expense_list.csv']
'''
```

19.3.3 데이터 처리

■ 자신이 직접 설계해서 만든 데이터가 아니고 외부에서 데이터를 가져온 경우 데이터가 원하는 구조가 아닐 수 있다. 이 경우 데이터 정제와 정리 과정이 필요하다.

(1) 수집된 데이터 파일의 구조 분석

■ 데이터 수집 이후에는 수집된 데이터에 문제가 없는지 검토하는 과정이 필요하다.
- 08~15: 텍스트 파일을 읽는 코드를 이용해 처음 세 줄을 읽어서 출력한다.
- 16~31: 데이터의 첫째 줄에는 열 이름이 있고, 둘째 줄부터는 데이터 값이 있다.
- 34~39: 첫째 줄에 있는 열 이름의 개수와 둘째 줄과 셋째 줄에 있는 데이터 값의 개수가 각각 몇 개인지 살펴본다.
- 46~56: "시정 충남, 전북 현장 방문 관련 업무협의"와 같이 따옴표 안의 콤마를 제거한다.

[ch19_project/ex06_데이터처리.py의 일부]

```
# 수집된 데이터 파일의 구조 분석
from datetime import datetime
import os
import pandas as pd
import glob
data_file = './ch19_project/seoul_expense/2017/201701_expense_list.csv'

with open(data_file, encoding='utf-8') as f:
    line1 = f.readline()
    line2 = f.readline()
    line3 = f.readline()

    print(line1)
    print(line2)
    print(line3)
```

```
'''
nid,title,url,dept_nm_lvl_1,dept_nm_lvl_2,dept_nm_lvl_3,dept_nm_lvl_4,dept_nm_lvl_5,exec_yr,exec_month,
expense_budget,expense_execution,category,dept_nm_full,exec_dt,exec_loc,exec_purpose,target_nm,payment_
method,exec_amount

11430252,"2017년 1월 장애인복지정책과 업무추진비 집행내역",http://opengov.seoul.go.kr/public/11430252,
서
울시본청,복지본부,장애인복지정책과,,,2017,1,,,,"복지본부 장애인복지정책과","2017-01-26 13:10","동해일식
(중구 무교동)","기본소득과 장애인복지 논의간담회","장애인복지정책팀장 외 2명",카드,76000

11430252,"2017년 1월 장애인복지정책과 업무추진비 집행내역",http://opengov.seoul.go.kr/public/11430252,
서
울시본청,복지본부,장애인복지정책과,,,2017,1,,,,"복지본부 장애인복지정책과","2017-01-25 22:41","김앤장 (
중구 무교로)","장애인단체 활동지원 논의간담회","장애인복지정책과장 외 3명",카드,102000

'''



line1_len = len(line1.split(','))
line2_len = len(line2.split(','))
line3_len = len(line3.split(','))

print("[각 줄의 데이터값의 개수]")
print("첫째 줄:{}, 둘째 줄:{}, 셋째 줄:{}".format(line1_len, line2_len, line3_len))
'''
[각 줄의 데이터값의 개수]
첫째 줄:20, 둘째 줄:20, 셋째 줄:20
'''



def get_value_count(line):

    line_rep_list = []
    for k, x in enumerate(line.split('"')):
        if(k % 2 != 0):
            x = x.replace(',', '')
        line_rep_list.append(x)

    line_rep_str = ''.join(line_rep_list)

    return len(line_rep_str.split(','))


line1_len = get_value_count(line1)
line2_len = get_value_count(line2)
line3_len = get_value_count(line3)

print("[각 줄의 데이터값의 개수]")
print("첫째 줄:{}, 둘째 줄:{}, 셋째 줄:{}".format(line1_len, line2_len, line3_len))
'''
[각 줄의 데이터값의 개수]
첫째 줄:20, 둘째 줄:20, 셋째 줄:20
'''
```

(2) 첫 번째 줄의 이름과 개수 변경

■ 서울시의 업무추진비 CSV 파일에 첫 번째 줄에 누락된 열 이름이 있는 경우, 열 이름을 추가하고 알아보기 어려운 열 이름은 한글로 바꾼다.

- 05: splitlines()는 문자열을 개행문자(\n)를 중심으로 나누는 함수이다.
- 13: '\n'.join(lines)는 lines에 있는 리스트 요소를 개행문자로 연결해서 하나의 문자

열로 만들어준다.

- 89~106: 모든 데이터 파일에 대해 첫 번째 줄의 열 이름을 변경한 파일이 잘 생성됐는지 확인한다.
- 102: 파이썬 내장 모듈인 glob.glob() 함수는 인자로 받은 패턴과 이름이 일치하는 모든 파일과 디렉터리의 리스트를 반환한다. 패턴을 그냥 *라고 주면 모든 파일과 디렉터리를 볼 수 있다.

[ch19_project/ex06_데이터처리.py의 일부]

```
# 2. 첫 번째 줄의 열 이름과 개수 변경
def change_csv_file_first_line_value(old_file_name, new_file_name):
    with open(old_file_name, encoding='utf-8') as f:  # 파일을 읽기 모드로 열기
        # 전체 데이터를 읽어서 한 줄씩 lines 리스트의 각 요소에 할당
        lines = f.read().splitlines()

    # 첫째 줄의 내용을 변경할 열 이름을 지정해서 변경
    lines[0] = 'nid,제목,url,부서레벨1,부서레벨2,부서레벨3,부서레벨4,부서레벨5,\
집행연도,집행월,예산,집행,구분,부서명,집행일시,집행장소,집행목적,대상인원,결제방법,집행금액'

    with open(new_file_name, 'w', encoding='utf-8') as f:  # 파일을 쓰기 모드로 열기
        # 리스트 내의 각 요소를 개행문자(\n)로 연결해서 파일로 저장
        f.write('\n'.join(lines))


# 기존의 파일
old_file_name = './ch19_project/seoul_expense/2017/201701_expense_list.csv'

# 새로운 파일
new_file_name = './ch19_project/seoul_expense/2017/201701_expense_list_new.csv'

# 첫째 줄의 내용을 변경한 새로운 파일 생성
change_csv_file_first_line_value(old_file_name, new_file_name)


with open(new_file_name, encoding='utf-8') as f:  # 파일을 읽기 모드로 열기
    for k in range(3):
        print(f.readline())
'''
nid,제목,url,부서레벨1,부서레벨2,부서레벨3,부서레벨4,부서레벨5,집행연도,집행월,예산,집행,구분,부서명,집
행일시
,집행장소,집행목적,대상인원,결제방법,집행금액

11430252,"2017년 1월 장애인복지정책과 업무추진비 집행내역",http://opengov.seoul.go.kr/public/11430252,
서울시
본청,복지본부,장애인복지정책과,,,2017,1,,,,"복지본부 장애인복지정책과","2017-01-26 13:10","동해일식 (중
구 무
교동)","기본소득과 장애인복지 논의간담회","장애인복지정책팀장 외 2명",카드,76000

11430252,"2017년 1월 장애인복지정책과 업무추진비 집행내역",http://opengov.seoul.go.kr/public/11430252,
서울시
본청,복지본부,장애인복지정책과,,,2017,1,,,,"복지본부 장애인복지정책과","2017-01-25 22:41","김앤장 (중구
무교
로)","장애인단체 활동지원 논의간담회","장애인복지정책과장 외 3명",카드,102000

'''


# 인자: 연도, 데이터 파일이 있는 폴더
def change_year_csv_file_first_line_value(year, data_folder):

    # 데이터 파일이 있는 폴더 지정
    # ex) 'C:/myPyCode/data/seoul_expense/2017/'
    expense_list_year_dir = data_folder + str(year) + '/'
```

```

    extension = 'csv'  # 확장자 이름

    # 지정한 폴더에 있는 월별 업무추진비 파일에서 첫 번째 줄의 열 이름을 변경
    for k in range(12):
        # 기존의 파일 이름 지정
        old_file_name = expense_list_year_dir + \
            '{0}{1:02d}_expense_list.{2}'.format(year, k+1, extension)

        # 새로운 파일 이름 지정
        new_file_name = expense_list_year_dir + \
            '{0}{1:02d}_expense_list_new.{2}'.format(year, k+1, extension)

        # 첫째 줄의 내용을 변경한 새로운 파일 생성
        change_csv_file_first_line_value(old_file_name, new_file_name)


data_folder = './ch19_project/seoul_expense/'

years = [2017, 2018, 2019]  # 연도를 지정

for year in years:
    print("{}년 데이터의 첫 번째 줄의 열 이름을 변경해서 새 파일에 저장합니다.".format(year))
    change_year_csv_file_first_line_value(year, data_folder)

print("모든 데이터의 첫 번째 줄의 열 이름을 변경해서 새 파일로 저장했습니다.")
'''
2017년 데이터의 첫 번째 줄의 열 이름을 변경해서 새 파일에 저장합니다.
2018년 데이터의 첫 번째 줄의 열 이름을 변경해서 새 파일에 저장합니다.
2019년 데이터의 첫 번째 줄의 열 이름을 변경해서 새 파일에 저장합니다.
모든 데이터의 첫 번째 줄의 열 이름을 변경해서 새 파일로 저장했습니다.
'''


data_folder = './ch19_project/seoul_expense/'

years = [2017, 2018, 2019]  # 연도를 지정

for year in years:
    path_name = data_folder + str(year)  # 폴더명을 지정
    print("[폴더 이름]", path_name)  # 폴더명 출력

    new_csv_files = []

    # 지정 폴더에서 파일명에 _new.csv가 포함된 파일만 지정
    file_name_for_glob = path_name + "/*_new.csv"

    for new_csv_file in glob.glob(file_name_for_glob):
        # 반환값에서 폴더는 제거하고 파일 이름만 추출
        new_csv_files.append(new_csv_file.split("\\")[-1])

    print("* 새롭게 생성된 CSV 파일:", new_csv_files)
'''
[폴더 이름] ./ch19_project/seoul_expense/2017
*  새롭게  생성된  CSV  파일:  ['201701_expense_list_new.csv',  '201702_expense_list_new.csv',
'201703_expense_list_new.csv',      '201704_expense_list_new.csv',      '201705_expense_list_new.csv',
'201706_expense_list_new.csv',      '201707_expense_list_new.csv',      '201708_expense_list_new.csv',
'201709_expense_list_new.csv',      '201710_expense_list_new.csv',      '201711_expense_list_new.csv',
'201712_expense_list_new.csv']
[폴더 이름] ./ch19_project/seoul_expense/2018
*  새롭게  생성된  CSV  파일:  ['201801_expense_list_new.csv',  '201802_expense_list_new.csv',
'201803_expense_list_new.csv',      '201804_expense_list_new.csv',      '201805_expense_list_new.csv',
'201806_expense_list_new.csv',      '201807_expense_list_new.csv',      '201808_expense_list_new.csv',
'201809_expense_list_new.csv',      '201810_expense_list_new.csv',      '201811_expense_list_new.csv',
'201812_expense_list_new.csv']
```

```
[폴더 이름] ./ch19_project/seoul_expense/2019
*   새롭게   생성된   CSV   파일:   ['201901_expense_list_new.csv',   '201902_expense_list_new.csv',
'201903_expense_list_new.csv',      '201904_expense_list_new.csv',      '201905_expense_list_new.csv',
'201906_expense_list_new.csv',      '201907_expense_list_new.csv',      '201908_expense_list_new.csv',
'201909_expense_list_new.csv',      '201910_expense_list_new.csv',      '201911_expense_list_new.csv',
'201912_expense_list_new.csv']
'''
```

(3) 데이터의 구조 및 결측치 살펴보기

- CSV 파일을 pandas의 DataFrame 형식으로 가져와서 데이터의 구조를 살펴보고 빠진 데이터 값(결측치)이 있는지도 살펴본다.
  - 68~79: isna()를 통해 결측치가 몇 개 있는지 확인한다.
  - 82~84: 결측치가 많은 열은 제거하고 나머지 열의 데이터만 이용한다.
  - 113~134: 원하는 열의 데이터만 선택해서 새로운 파일로 저장한다.
  - 137~144: 2017년, 2018년, 2019년에 해당하는 모든 데이터를 읽어서 필요 없는 열은 제거하고 새로운 파일 이름(연도_expense_list_tidy.csv)으로 저장한다.

[ch19_project/ex06_데이터처리.py의 일부]

```
# 3. 데이터의 구조 및 결측시 살펴보기
expense_list2016_dir = './ch19_project/seoul_expense/2017/'
file_name = "201701_expense_list_new.csv"

df = pd.read_csv(expense_list2016_dir + file_name)
print(df.head(2))
'''
        nid                            제목  ... 결제방법    집행금액
0  11430252  2017년 1월 장애인복지정책과 업무추진비 집행내역  ...   카드   76000
1  11430252  2017년 1월 장애인복지정책과 업무추진비 집행내역  ...   카드  102000

[2 rows x 20 columns]
'''


year = 2017
expense_list_year_dir = './ch19_project/seoul_expense/' + str(year) + '/'

df_year = pd.DataFrame()
for k in range(12):

    # 파일 이름 지정
    file_name = "{0}{1:02d}_expense_list_new.csv".format(year, k+1)

    # pandas DataFrame 형식으로 csv 데이터 불러오기
    df_month = pd.read_csv(expense_list_year_dir + file_name)

    # df_year에 df_month를 세로 방향으로 추가해서 다시 df_year에 할당
    # 통합된 dataFrame의 순서대로 index를 할당하기 위해서 `ignore_index = True` 옵션 지정
    df_year = df_year.append(df_month, ignore_index=True)

print(df_year.head(2))
'''
        nid                            제목  ... 결제방법    집행금액
0  11430252  2017년 1월 장애인복지정책과 업무추진비 집행내역  ...   카드   76000
1  11430252  2017년 1월 장애인복지정책과 업무추진비 집행내역  ...   카드  102000

[2 rows x 20 columns]
'''
```

```
40
41  print(df_year.tail(2))
42  '''
43              nid                                제목  ... 결제방법    집행금액
44  70130  14292506  2017년 12월 사업소_은평병원_원무과 업무추진비 내역  ...   카드  820000
45  70131  14292506  2017년 12월 사업소_은평병원_원무과 업무추진비 내역  ...   카드  440000
46
47  [2 rows x 20 columns]
48  '''
49
50  print(df_year.info())
51  '''
52  <class 'pandas.core.frame.DataFrame'>
53  RangeIndex: 70132 entries, 0 to 70131
54  Data columns (total 20 columns):
55   #   Column  Non-Null Count  Dtype
56  ---  ------  --------------  -----
57   0   nid     70132 non-null  int64
58   1   제목      70132 non-null  object
59   2   url     70132 non-null  object
60   ...(생략)...
61   18  결제방법    69929 non-null  object
62   19  집행금액    70132 non-null  int64
63  dtypes: float64(2), int64(4), object(14)
64  memory usage: 10.7+ MB
65  None
66  '''
67
68  print(df_year.isna().sum())
69  '''
70  nid          0
71  제목           0
72  url          0
73  부서레벨1        0
74  부서레벨2       58
75  ...(생략)...
76  결제방법       203
77  집행금액         0
78  dtype: int64
79  '''
80
81
82  df_year_drop = df_year.drop(columns=['nid', 'url', '부서레벨3', '부서레벨4', '부서레벨5',
83                                       '예산', '집행', '구분'])
84  print(df_year_drop.head(2))
85  '''
86                              제목   부서레벨1  부서레벨2   집행연도  ...                    집행목적
87  대상
88  인원 결제방법    집행금액
89  0  2017년 1월 장애인복지정책과 업무추진비 집행내역  서울시본청  복지본부  2017  ...  기본소득과 장애인
90  복지 논
91  의간담회  장애인복지정책팀장 외 2명   카드   76000
92  1  2017년 1월 장애인복지정책과 업무추진비 집행내역  서울시본청  복지본부  2017  ...   장애인단체 활동지
93  원 논
94  의간담회  장애인복지정책과장 외 3명   카드  102000
95
96  [2 rows x 12 columns]
97  '''
98
99
100 year = 2017
101 expense_list_year_dir = './ch19_project/seoul_expense/' + str(year) + '/'
102
103 expense_list_tidy_file = "{}_expense_list_tidy.csv".format(year)
104 df_year_drop.to_csv(expense_list_year_dir +
```

```
                    expense_list_tidy_file, index=False)


file_name = expense_list_year_dir + expense_list_tidy_file
print(file_name)
os.path.isfile(file_name)


def select_columns_save_file(year, data_folder, drop_columns_list):

    expense_list_year_dir = data_folder + str(year) + '/'
    expense_list_tidy_file = "{}_expense_list_tidy.csv".format(year)
    df_year = pd.DataFrame()

    for k in range(12):
        # 파일 이름 지정
        file_name = "{0}{1:02d}_expense_list_new.csv".format(year, k+1)

        # aDtaFrame 형식으로 csv 데이터 불러오기
        df_month = pd.read_csv(expense_list_year_dir + file_name)

        # fd_year에 df_month를 새로 추가해서 다시 df_year에 할당
        # 통합된 adtaFrame의 순서대로 index를 할당하기 위해서 `ignore_index = True` 옵션 지정
        df_year = df_year.append(df_month, ignore_index=True)

    df_year_drop = df_year.drop(columns=drop_columns_list)
    new_file_name = expense_list_year_dir + expense_list_tidy_file
    df_year_drop.to_csv(new_file_name, index=False)

    print("==> {} 파일을 생성했습니다.".format(expense_list_tidy_file))


data_folder = './ch19_project/seoul_expense/'
years = [2017, 2018, 2019]
drop_columns_list = ['nid', 'url', '부서레벨3', '부서레벨4', '부서레벨5', '예산', '집행', '구분']

for year in years:
    print("{}년 데이터를 정리해서 저장하고 있습니다.".format(year))
    select_columns_save_file(year, data_folder, drop_columns_list)
print("모든 연도의 데이터를 정리해서 파일로 저장했습니다.")
'''
./ch19_project/seoul_expense/2017/2017_expense_list_tidy.csv
2017년 데이터를 정리해서 저장하고 있습니다.
==> 2017_expense_list_tidy.csv 파일을 생성했습니다.
2018년 데이터를 정리해서 저장하고 있습니다.
==> 2018_expense_list_tidy.csv 파일을 생성했습니다.
2019년 데이터를 정리해서 저장하고 있습니다.
==> 2019_expense_list_tidy.csv 파일을 생성했습니다.
모든 연도의 데이터를 정리해서 파일로 저장했습니다.
'''

years = [2017, 2018, 2019]

for year in years:

    expense_list_year_dir = data_folder + str(year) + '/'
    expense_list_tidy_file = "{}_expense_list_tidy.csv".format(year)

    file_name = expense_list_year_dir + expense_list_tidy_file
    print(file_name, "==> ", end="")
    print(os.path.isfile(file_name))


def get_file_info(year, data_folder):
    expense_list_year_dir = data_folder + str(year) + '/'
```

```
    expense_list_tidy_file = "{}_expense_list_tidy.csv".format(year)

    path_file_name = expense_list_year_dir + expense_list_tidy_file
    print(path_file_name)
    result = os.path.isfile(path_file_name)

    # 파일 수정 시간
    modified_time = datetime.fromtimestamp(os.path.getmtime(path_file_name))

    # 파일 생성 시간
    created_time = datetime.fromtimestamp(os.path.getctime(path_file_name))

    # 파일 크기
    file_size = os.path.getsize(path_file_name)

    if(result == True):
        print("[생성한 CSV 데이터 파일의 정보]")
        print('* 폴더 위치 :', expense_list_year_dir)
        print('* 파일 이름 :', expense_list_tidy_file)
        print('* 수정 시간 :', modified_time.strftime('%Y-%m-%d %H:%M:%S'))
        print('* 생성 시간 :', created_time.strftime('%Y-%m-%d %H:%M:%S'))
        print('* 파일 크기 : {0:,} 바이트'.format(file_size))


data_folder = './ch19_project/seoul_expense/'
years = [2017, 2018, 2019]

for year in years:

    get_file_info(year, data_folder)
    print("")
'''
[생성한 CSV 데이터 파일의 정보]
* 폴더 위치 : ./ch19_project/seoul_expense/2017/
* 파일 이름 : 2017_expense_list_tidy.csv
* 수정 시간 : 2021-11-21 11:22:37
* 생성 시간 : 2021-11-20 23:27:26
* 파일 크기 : 21,505,565 바이트

./ch19_project/seoul_expense/2018/2018_expense_list_tidy.csv
[생성한 CSV 데이터 파일의 정보]
* 폴더 위치 : ./ch19_project/seoul_expense/2018/
* 파일 이름 : 2018_expense_list_tidy.csv
* 수정 시간 : 2021-11-21 11:22:38
* 생성 시간 : 2021-11-20 23:32:30
* 파일 크기 : 23,012,590 바이트

./ch19_project/seoul_expense/2019/2019_expense_list_tidy.csv
[생성한 CSV 데이터 파일의 정보]
* 폴더 위치 : ./ch19_project/seoul_expense/2019/
* 파일 이름 : 2019_expense_list_tidy.csv
* 수정 시간 : 2021-11-21 11:22:39
* 생성 시간 : 2021-11-20 23:32:31
* 파일 크기 : 24,459,405 바이트
'''
```

19.3.4 데이터 분석

- 2017년, 2018년, 2019년 서울시의 업무추진비 데이터를 가져와 정제한 후 필요한 데이터만 남겨서 CSV 파일로 저장했다. 이제 이 파일을 이용해 서울시의 업무추진비가 어떻게 집행됐는지 분석해 본다.

- 07~19: 정제 후 깔끔하게 정리된 2017년, 2018년, 2019년치 데이터 파일을 DataFrame 형식으로 가져와 하나로 통합한다.
- 22: df_expense_all의 전체 데이터 구조를 확인하기 위해 info()를 사용한다.
- 23~44: 3년간의 업무추진비 전체 데이터 파일에서 데이터를 읽어도다 보니 데이터 개수가 아주 많다.
- 46~59: df_expense_all 변수의 앞쪽 데이터 일부를 확인해 본다.
- 61~74: df_expense_all 변수의 뒤쪽 데이터 일부도 확인해 본다.

[ch19_project/ex07_데이터분석.py의 일부]

```
01  # 데이터 분석
02  from wordcloud import WordCloud
03  import matplotlib
04  import matplotlib.pyplot as plt
05  import pandas as pd
06
07  data_folder = './ch19_project/seoul_expense/'
08  years = [2017, 2018, 2019]
09
10  df_expense_all = pd.DataFrame()
11
12  for year in years:
13      expense_list_year_dir = data_folder + str(year) + '/'
14      expense_list_tidy_file = "{}_expense_list_tidy.csv".format(year)
15
16      path_file_name = expense_list_year_dir + expense_list_tidy_file
17
18      df_expense = pd.read_csv(path_file_name)
19      df_expense_all = df_expense_all.append(df_expense, ignore_index=True)
20
21
22  print(df_expense_all.info())
23  '''
24  <class 'pandas.core.frame.DataFrame'>
25  RangeIndex: 216557 entries, 0 to 216556
26  Data columns (total 12 columns):
27   #   Column  Non-Null Count   Dtype
28  ---  ------  --------------   -----
29   0   제목      216557 non-null  object
30   1   부서레벨1   216557 non-null  object
31   2   부서레벨2   216273 non-null  object
32   3   집행연도    216557 non-null  int64
33   4   집행월     216557 non-null  int64
34   5   부서명     216478 non-null  object
35   6   집행일시    216557 non-null  object
36   7   집행장소    214401 non-null  object
37   8   집행목적    216535 non-null  object
38   9   대상인원    215535 non-null  object
39   10  결제방법    216354 non-null  object
40   11  집행금액    216557 non-null  int64
41  dtypes: int64(3), object(9)
42  memory usage: 19.8+ MB
43  None
44  '''
45
46  print(df_expense_all.head(2))
47  '''
48                              제목  부서레벨1 부서레벨2  집행연도  ...                    집행목적
49  대상
50  인원 결제방법    집행금액
51  0  2017년 1월 장애인복지정책과 업무추진비 집행내역  서울시본청  복지본부  2017  ...  기본소득과 장애인
52  복지 논
```

```
53  의간담회  장애인복지정책팀장 외 2명   카드   76000
54  1  2017년 1월 장애인복지정책과 업무추진비 집행내역  서울시본청  복지본부  2017  ...   장애인단체 활동지
55  원 논
56  의간담회  장애인복지정책과장 외 3명   카드  102000
57
58  [2 rows x 12 columns]
59  '''
60
61  print(df_expense_all.tail(2))
62  '''
63                                                제목 부서레벨1    부서레벨2  ...       대상인원  결제방
64  법
65  집행금액
66  216555  2019년 12월 사업소 서울시립대학교 중앙도서관 사서과 업무추진비 - 전체   사업소  서울시립대학
67  교  ...
68  사서과장 외 8명    카드  23940
69  216556  2019년 12월 사업소 서울시립대학교 중앙도서관 사서과 업무추진비 - 전체   사업소  서울시립대학
70  교  ...
71    이용자 80명    카드  53420
72
73  [2 rows x 12 columns]
74  '''
```

(1) 연도별 추이 분석

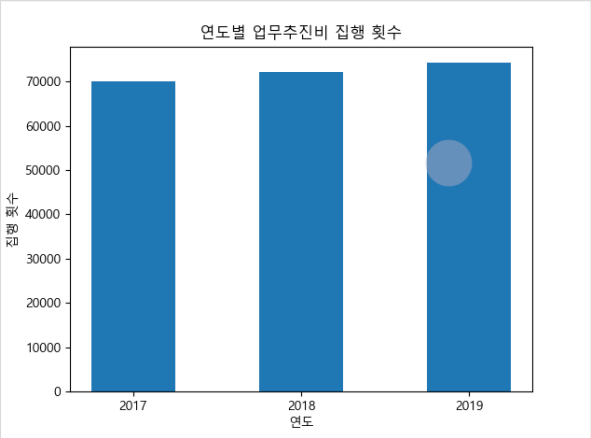
■ 2017년부터 2019년까지 연도별 전체 집행 금액을 비교해서 추이 분석을 해 본다.

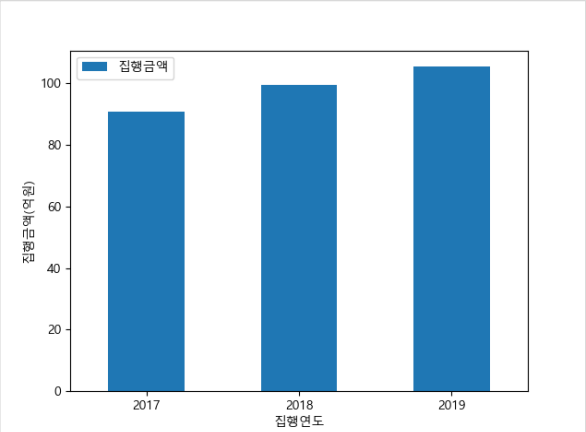
- 02: 연도별 업무추진비 집행 횟수를 알고 싶다면 '집행연도' 열에 대해 value_counts()를 수행한다.
- 12~20: 앞의 데이터를 막대 그래프로 시각화한다.

[ch19_project/ex07_데이터분석.py의 일부]

```
01  # 1. 연도별 추이 분석
02  year_expense = df_expense_all['집행연도'].value_counts()
03  print(year_expense)
04  '''
05  2019    74207
06  2018    72218
07  2017    70132
08  Name: 집행연도, dtype: int64
09  '''
10
11  matplotlib.rcParams['font.family'] = 'Malgun Gothic'
12  matplotlib.rcParams['axes.unicode_minus'] = False
13
14  plt.bar(year_expense.index, year_expense.values,
15          tick_label=year_expense.index, width=0.5)
16  plt.title("연도별 업무추진비 집행 횟수")
17  plt.xlabel("연도")
18  plt.ylabel("집행 횟수")
19  plt.show()
20
21
22  year_total = pd.pivot_table(df_expense_all, index=[
23                              '집행연도'], values=['집행금액'], aggfunc=sum)
24  print(year_total)
25  '''
26              집행금액
27  집행연도
28  2017   9076941387
```

```
2018   9937556542
2019  10532330632
'''

eok_won = 100000000  # 억원
(year_total/eok_won).plot.bar(rot=0)  # 'rot = 각도'로 xtick 회전 각도를 지정
plt.ylabel('집행금액(억원)')
plt.show()
```





(2) 월별 집행금액 분석

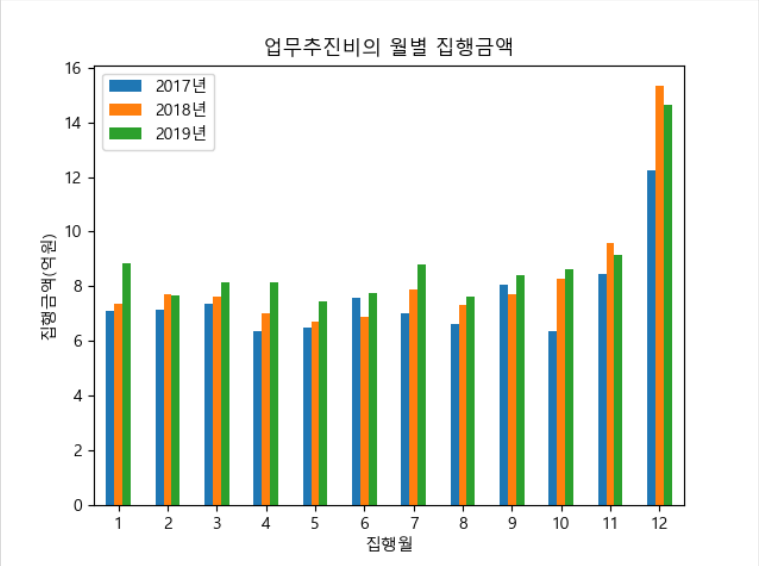
■ 이번에는 월별로 합계를 구해서 1월에서 12월까지 월별로 업무추진비 집행금액의 변화를 살펴본다.

- 02: pivot_table()을 이용해 집행월별 집행금액의 합을 구한다.
- 22~23: 3년간의 데이터에서 월별 집행금액만 추출한다.

[ch19_project/ex07_데이터분석.py의 일부]

```
# 2. 월별 집행금액 분석
month_total = pd.pivot_table(df_expense_all, index=['집행월'], values=['집행금액'],
                             aggfunc=sum)
print(month_total)
'''
          집행금액
집행월
1    2328469179
2    2250971737
3    2314589911
4    2153704599
5    2063883588
6    2224855495
7    2372256669
8    2153716469
9    2417217365
10   2326108698
11   2719921484
12   4221133367
'''

year_month_total = pd.pivot_table(df_expense_all, index=['집행월'], columns=['집행연도'],
                                  values=['집행금액'], aggfunc=sum)
print(year_month_total)
'''
```

```
                집행금액
집행연도          2017         2018         2019
집행월
1       710368860   735587570   882512749
2       712679864   769360005   768931868
3       737250454   761059010   816280447
4       635265805   703781418   814657376
5       647582378   669044701   747256509
6       758257342   690652154   775945999
7       701604626   788926477   881725566
8       661174850   730290532   762251087
9       806170700   769404957   841641708
10      637219943   827022975   861865780
11      843619171   960310221   915992092
12     1225747394  1532116522  1463269451
'''

eok_won = 100000000  # 억원

(year_month_total/eok_won).plot.bar(rot=0)
plt.ylabel('집행금액(억원)')
plt.title("업무추진비의 월별 집행금액")
plt.legend(['2017년', '2018년', '2019년'])
plt.show()
```



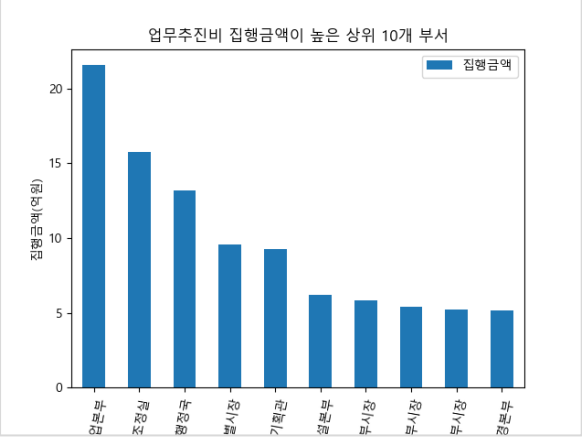
(3) 부서별 집행 내역 분석

■ 이번에는 부서별로 업무 추진비 집행 내역을 분석해 본다.

[ch19_project/ex07_데이터분석.py의 일부]

```
# 3. 부서별 집행 내역 분석
dept_level1_total = pd.pivot_table(df_expense_all, index=['부서레벨1'], values=['집행금액'],
                                   aggfunc=sum)
print(dept_level1_total)
'''
                  집행금액
부서레벨1
사업소          6552128899
```

```
서울시본청        16606242519
소방재난본부(소방서)   5147645293
의회사무처         1240811850
'''


dept_level_2_total = pd.pivot_table(df_expense_all, index=['부서레벨2'], values=['집행금액'],
                                    aggfunc=sum)
print(dept_level_2_total.head())
'''
               집행금액
부서레벨2
119특수구조단  119225100
감사위원회     343281170
강남소방서     229660520
강동소방서     188773330
강북소방서     167700000
'''


dept_level_2_total_top10 = dept_level_2_total.sort_values(
    by=['집행금액'], ascending=False)[0:10]
print(dept_level_2_total_top10)
'''
                집행금액
부서레벨2
상수도사업본부   2156404778
기획조정실     1572753168
행정국       1320839804
서울특별시장     955448760
시민소통기획관    923338423
도시기반시설본부   620669144
정무부시장      581806882
행정1부시장     540457390
행정2부시장     522277598
기후환경본부     515222890
'''

eok_won = 100000000  # 억원

(dept_level_2_total_top10/eok_won).plot.bar(rot=80)
plt.ylabel('집행금액(억원)')
plt.title("업무추진비 집행금액이 높은 상위 10개 부서")
plt.show()


korean_font_path = 'C:/Windows/Fonts/malgun.ttf'  # 한글 폰트(맑은 고딕) 파일명

# 워드 클라우드 이미지 생성
wc = WordCloud(font_path=korean_font_path, background_color='white',
               width=800, height=600)

frequencies = dept_level_2_total['집행금액']  # pandas의 Series 형식이 됨
wordcloud_image = wc.generate_from_frequencies(frequencies)

plt.figure(figsize=(12, 9))
plt.axis('off')
plt.imshow(wordcloud_image, interpolation='bilinear')
plt.show()
```



(4) 요일별 및 시간대별 집행 내역 분석

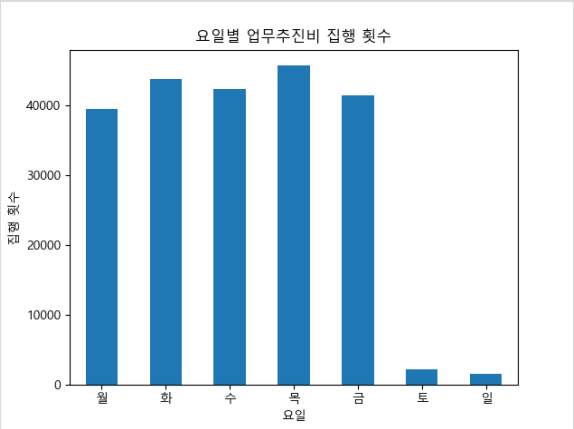
■ 이번에는 '집행일시'열에 있는 날짜와 시간을 이용해 요일별/시간대별로 업무추진비 진행 내역을 분석한다. 요일별/시간대별로 분석하면 무슨 요일, 무슨 시간대에 업무추진비를 많이 집행했는지 알 수 있다.

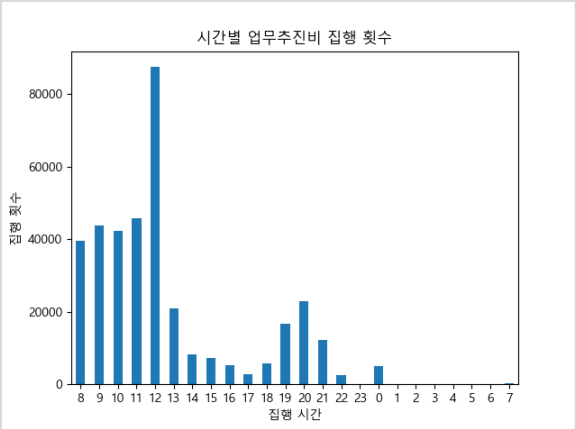
[ch19_project/ex07_데이터분석.py의 일부]

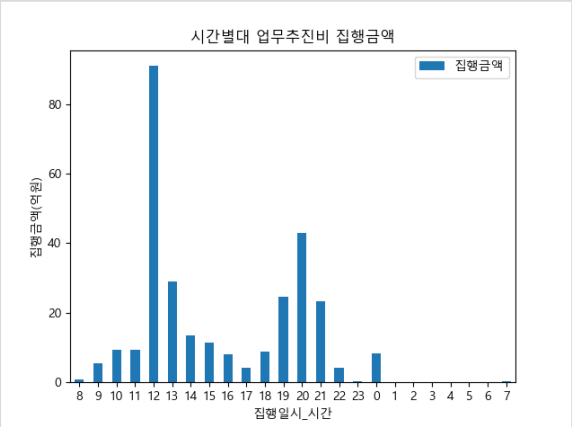
```
# 4. 요일별 및 시간대별 집행 내역 분석
print(df_expense_all['집행일시'].values)
'''
['2017-01-26 13:10' '2017-01-25 22:41' '2017-01-24 12:35' ...
 '2019-12-19 11:34' '2019-12-16 12:39' '2019-12-03 17:35']
'''

expense_date_time = pd.to_datetime(df_expense_all['집행일시'])
print(expense_date_time.values)
'''
['2017-01-26T13:10:00.000000000' '2017-01-25T22:41:00.000000000'
 '2017-01-24T12:35:00.000000000' ... '2019-12-19T11:34:00.000000000'
 '2019-12-16T12:39:00.000000000' '2019-12-03T17:35:00.000000000']
'''

week_day_name = ["월", "화", "수", "목", "금", "토", "일"]

df_expense_all['집행일시_요일'] = [week_day_name[weekday]
                              for weekday in expense_date_time.dt.weekday]

df_expense_all['집행일시_시간'] = [hour for hour in expense_date_time.dt.hour]

print(df_expense_all.head(3))
'''
  집행금액 집행일시_요일 집행일시_시간
0  2017년 1월 장애인복지정책과 업무추진비 집행내역  서울시본청  복지본부  2017    1  ...  장애인복지정
책팀장
외 2명   카드   76000       목      13
1  2017년 1월 장애인복지정책과 업무추진비 집행내역  서울시본청  복지본부  2017    1  ...  장애인복지정
책과장
외 3명   카드  102000       수      22
2  2017년 1월 장애인복지정책과 업무추진비 집행내역  서울시본청  복지본부  2017    1  ...    장애인복지
정책팀
장외7명   카드   80000       화      12

[3 rows x 14 columns]
'''
```

```

expense_weekday = df_expense_all['집행일시_요일'].value_counts()
print(expense_weekday)
'''
목    45683
화    43812
수    42343
금    41381
월    39498
토     2238
일     1602
Name: 집행일시_요일, dtype: int64
'''

expense_weekday = expense_weekday.reindex(index=week_day_name)
print(expense_weekday)
'''
월    39498
화    43812
수    42343
목    45683
금    41381
토     2238
일     1602
Name: 집행일시_요일, dtype: int64
'''

expense_weekday.plot.bar(rot=0)
plt.title("요일별 업무추진비 집행 횟수")
plt.xlabel("요일")
plt.ylabel("집행 횟수")
plt.show()


expense_hour_num = df_expense_all['집행일시_시간'].value_counts()
print(expense_hour_num)
'''
12    87518
20    23013
13    20990
19    16766
21    12210
11     8356
14     8311
15     7168
10     5824
18     5783
16     5169
0      4919
9      3486
17     2889
22     2563
8       875
7       412
23      128
1        44
6        42
3        27
4        26
5        19
2        19
Name: 집행일시_시간, dtype: int64
'''

```

```
103
104 work_hour = [(k+8) % 24 for k in range(24)]
105 expense_hour_num = expense_hour_num.reindex(index=work_hour)
106 print(expense_hour_num)
107 '''
108 8       875
109 9      3486
110 10     5824
111 11     8356
112 12    87518
113 13    20990
114 14     8311
115 15     7168
116 16     5169
117 17     2889
118 18     5783
119 19    16766
120 20    23013
121 21    12210
122 22     2563
123 23      128
124 0      4919
125 1        44
126 2        19
127 3        27
128 4        26
129 5        19
130 6        42
131 7       412
132 Name: 집행일시_시간, dtype: int64
133 '''
134
135
136 expense_hour_num.plot.bar(rot=0)
137 plt.title("시간별 업무추진비 집행 횟수")
138 plt.xlabel("집행 시간")
139 plt.ylabel("집행 횟수")
140 # plt.show()
141
142
143 expense_hour_total = pd.pivot_table(df_expense_all, index=['집행일시_시간'],
144                                     values=['집행금액'], aggfunc=sum)
145 print(expense_hour_total.head())
146 '''
147 집행일시_시간
148 0        842523116
149 1          7024161
150 2          2265190
151 3          7215762
152 4          5818431
153 '''
154
155
156 eok_won = 100000000  # 억원
157 expense_hour_total = expense_hour_total.reindex(index=work_hour)
158
159 (expense_hour_total/eok_won).plot.bar(rot=0)
160 plt.ylabel('집행금액(억원)')
161 plt.title("시간별대 업무추진비 집행금액")
162 plt.show()
```


요일별 업무추진비 집행 횟수
집행 횟수
0
10000
20000
30000
40000
월
화
수
목
금
토
일
요일


시간별 업무추진비 집행 횟수
집행 횟수
0
20000
40000
60000
80000
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0 1 2 3 4 5 6 7
집행 시간


시간별대 업무추진비 집행금액
집행금액
집행금액(억원)
0
20
40
60
80
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0 1 2 3 4 5 6 7
집행일시_시간