국토교통 빅데이터 온라인 해커톤 경진대회

주제 : 포스트 코로나, 고령자의 안전한 이동을 위한 서울시 자치구 및 지하철역 분석

목차 :

필요한 모듈 가져오기

```
import pandas as pd
import json
#!pip install folium
import folium
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

#plotly
import plotly
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from plotly import tools

# for offline ploting
import plotly.offline as offline

# cufflinks
import cufflinks as cf
cf.go_offline()

# for folium plugins(지도 및 마커 시각화)
import os
import folium
from folium import plugins
print(folium.__version__)

#googlemap
import googlemaps

#json 파일 불러오기
import json

# matplotlib 폰트 변경(한글 사용)
import platform

from matplotlib import font_manager, rc
plt.rcParams['axes.unicode_minus'] = False

if platform.system() == 'Darwin':
    rc('font', family='AppleGothic')
elif platform.system() == 'Windows':
    path = "c:/Windows/Fonts/malgun.ttf"
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
```

```
else:
    print('Unknown system')
```

```
# 첫 번째 40,000,000 rows 분할해서 불러오기
df_card_1 = pd.read_csv('../data/DM_PUBTRF_USESTF_T.dat', skiprows = range(0, 0), nrows=4e7, sep='|', header=None, )
columns = ['년도','년월','운행 일자','요일 구분','이용자 유형 코드','정산사 ID',
           '정산 지역 코드','교통수단 구분 코드','노선 ID','정류장 ID','시도 코드',
          '시군구 코드', '이용 지역 코드', '이용 인원']
df_card_1.columns = columns

# 두 번째 40,000,000 rows 분할해서 불러오기
df_card_2 = pd.read_csv('../data//DM_PUBTRF_USESTF_T.dat', skiprows = range(0, 40000001), nrows=4e7, sep='|', header=None, )
df_card_2.columns = columns

# 세 번째 40,000,000 rows 분할해서 불러오기
df_card_3 = pd.read_csv('../data/DM_PUBTRF_USESTF_T.dat', skiprows = range(0, 80000001), nrows=4e7, sep='|', header=None, )
df_card_3.columns = columns

# 네 번째 40,000,000 rows 분할해서 불러오기
df_card_4 = pd.read_csv('../data/DM_PUBTRF_USESTF_T.dat', skiprows = range(0, 120000001), nrows=3e7, sep='|', header=None, )
df_card_4.columns = columns
```

이용자 유형 데이터-경로자 코드 확인

```
# 데이터 불러오기
user_type = pd.read_csv('../data/DW_USER_TYPE.dat',
                        encoding='utf-8',sep='|',header=None)

# 컬럼이름 정해주고, 인덱스 행 설정
user_type.columns=['code','user_type']
user_type.set_index('code')
```

지역 코드 데이터 확인 - 시군구

```
# 데이터 불러오기
area = pd.read_csv('../data/DD_AREA.dat',encoding='utf-8',sep='|')

# 컬럼명 변경
area.columns = ['지역구분','시도 코드','시군구 코드','이용지역 코드','시도','시군구','읍면동']

# 구별 지역코드 추출
# '시도'열이 서울특별시인 행만 추출
area_code_seoul = area[area['시도']=='서울특별시']

# 구만 나오는 행 추출
area_code_seoul = area_code_seoul[area_code_seoul['시군구']==area_code_seoul['읍면동']]

# 필요하지 않는 열 제거 - 지역구분, 이용지역 코드, 읍면동
area_code_seoul.drop(['지역구분','이용지역 코드','읍면동'], axis=1, inplace=True)

# 인덱스 정리
# 필요한 지역코드 데이터 추출 완료
area_code_seoul.reset_index(drop=True, inplace=True)
area_code_seoul.head()

# 시도 코드 확인 결과 : 서울특별시(11)
```

필요한 행 추출

```
# 필요한 조건의 행 추출하기
# - 이용자 유형 코드 : 경로(4)
# - 시도 코드 : 서울특별시(11)
# - 교통수단 구분 코드 : 지하철(T)
senior_card_1 = df_card_1[(df_card_1['이용자 유형 코드']==4) & (df_card_1['교통수단 구분 코드']=='T') & (df_card_1['시도 코드']==11)]
senior_card_2 = df_card_2[(df_card_2['이용자 유형 코드']==4) & (df_card_2['교통수단 구분 코드']=='T') & (df_card_2['시도 코드']==11)]
senior_card_3 = df_card_3[(df_card_3['이용자 유형 코드']==4) & (df_card_3['교통수단 구분 코드']=='T') & (df_card_3['시도 코드']==11)]
```

```
senior_card_4 = df_card_4[(df_card_4['이용자 유형 코드']==4) & (df_card_4['교통수단 구분 코드']=='T') & (df_card_4['시도 코드']==11)]


# 데이터 병합
senior_card_sub = pd.concat([senior_card_1, senior_card_2, senior_card_3, senior_card_4])

# 열 정리 : '년도', '년월', '시도 코드', '시군구 코드', '이용 인원'
senior_card_sub.drop(['이용자 유형 코드','정산사 ID','정산 지역 코드',
                      '교통수단 구분 코드','노선 ID','정류장 ID','이용 지역 코드'], axis=1, inplace=True)

# 인덱스 정리
senior_card_sub.reset_index(drop=True,inplace=True)
senior_card_sub.head()

# 지하철이용인원 데이터에 지역코드 데이터 merge()
# 사용 데이터 : 지하철 이용인원 데이터(senior_card_sub), 지역코드 데이터(DD.AREA.dat)

# 데이터 병합
senior_card_sub = pd.merge(area_code_seoul, senior_card_sub, how='outer',on=['시도 코드','시군구 코드'])

# 시군구코드 == 11000 인 행 제거(서울특별시 | 서울특별시 : 11000)
senior_card_sub = senior_card_sub[senior_card_sub['시군구 코드'] != 11000]
senior_card_sub
```

1. 코로나가 서울시 대중교통에 미친 영향

버스 vs 지하철 코로나 전후 증감률 비교

```
# 전 연령 버스 이용 증감률 계산하기 위한 새 변수 생성: 서울특별시(11), 버스(B)
allage_bus_1 = df_card_1[(df_card_1['교통수단 구분 코드']=='B') & (df_card_1['시도 코드']==11)][['년도','년월','이용자 유형 코드','시군구 코드','이용 인원'
allage_bus_2 = df_card_2[(df_card_2['교통수단 구분 코드']=='B') & (df_card_2['시도 코드']==11)][['년도','년월','이용자 유형 코드','시군구 코드','이용 인원'
allage_bus_3 = df_card_3[(df_card_3['교통수단 구분 코드']=='B') & (df_card_3['시도 코드']==11)][['년도','년월','이용자 유형 코드','시군구 코드','이용 인원'
allage_bus_4 = df_card_4[(df_card_4['교통수단 구분 코드']=='B') & (df_card_4['시도 코드']==11)][['년도','년월','이용자 유형 코드','시군구 코드','이용 인원'

# 전 연령 지하철 이용 증감률 계산하기 위한 새 변수 생성:서울특별시(11), 지하철(T)
allage_card_1 = df_card_1[(df_card_1['교통수단 구분 코드']=='T') & (df_card_1['시도 코드']==11)][['년도','년월','이용자 유형 코드','시군구 코드','이용 인원
allage_card_2 = df_card_2[(df_card_2['교통수단 구분 코드']=='T') & (df_card_2['시도 코드']==11)][['년도','년월','이용자 유형 코드','시군구 코드','이용 인원
allage_card_3 = df_card_3[(df_card_3['교통수단 구분 코드']=='T') & (df_card_3['시도 코드']==11)][['년도','년월','이용자 유형 코드','시군구 코드','이용 인원
allage_card_4 = df_card_4[(df_card_4['교통수단 구분 코드']=='T') & (df_card_4['시도 코드']==11)][['년도','년월','이용자 유형 코드','시군구 코드','이용 인원

# 데이터 병합 : allage_card_bus, allage_card_sub
allage_card_bus= pd.concat([allage_bus_1, allage_bus_2, allage_bus_3, allage_bus_4])
allage_card_sub= pd.concat([allage_card_1, allage_card_2, allage_card_3, allage_card_4])

# 전 연령 년도별 버스 이용 인원 합계
allage_bus_total = allage_card_bus.pivot_table('이용 인원','년도',aggfunc='sum')
allage_bus_total

# 버스 증감률 계산(2020-2019)/2019*100
allage_bus_total.reset_index(drop=True,inplace=True)
allbusupdown=round((allage_bus_total['이용 인원'][1]-allage_bus_total['이용 인원'][0])/allage_bus_total['이용 인원'][0]*100,2)
allbusupdown

# 전 연령 년도별 지하철 이용 인원 합계
allage_card_sub1= allage_card_sub.pivot_table('이용 인원','년도',aggfunc='sum')
allage_card_sub1

#지하철 증감률 계산(2020-2019)/2019*100
allage_card_sub1.reset_index(drop=True,inplace=True)
allsubupdown=round((allage_card_sub1['이용 인원'][1]-allage_card_sub1['이용 인원'][0])/allage_card_sub1['이용 인원'][0]*100,2)
allsubupdown

# 코로나 전,후 월 별 전연령 버스 이용 증감률 구하기
allage_bus_sub_updown=allage_card_bus.pivot_table('이용 인원','년월','년도', aggfunc='sum')
allage_bus_sub_updown

#버스 2019년 대비 2020년 증감률 구하기

b2019 = [0, 0, 0, 0, 0]
b19 = [58266908.0,50201719.0,60822158.0,60653528.0,62107587.0]
b29 = [54546466.0,47107718.0,40586588.0,42281059.0,46510338.0]
for i, (b19i, b29i) in enumerate(zip(b19, b29)):
    b2019[i] = round((b29i - b19i)/b19i*100,2)

b2019

# 코로나 전,후 월 별 전연령 지하철 이용 증감률 구하기
allage_card_sub_updown=allage_card_sub.pivot_table('이용 인원','년월','년도', aggfunc='sum')
```

```
allage_card_sub_updown

#지하철 2019년 대비 2020년 증감률 구하기

s2019 = [0, 0, 0, 0, 0]
a19 = [60217071.0,52082904.0,62617478.0,63089595.0,65066408.0]
a29 = [57351066.0,46678500.0,39054737.0,41718144.0,46327637.0]
for i, (a19i, a29i) in enumerate(zip(a19, a29)):
    s2019[i] = round((a29i - a19i)/a19i*100,2)

s2019
```

2. 코로나 데이터 분석을 통한 고령층의 코로나 치명성 검증

```
# 데이터 불러오기 : covid_TimeAge.csv
covid_TimeAge = pd.read_csv('../data/TimeAge.csv')

# 컬럼명 변경 : 테이블 정의서 참고
covid_TimeAge.columns = ['날짜','시간','환자의 나이','누적 확진자 수','누적 사망자 수']

# 불필요한 컬럼 제거
covid_TimeAge.drop(['시간'], axis=1, inplace=True)

covid_TimeAge.head()

# 마지막 집계일(2020.06.30)만 추출
total_0630 = covid_TimeAge[covid_TimeAge['날짜']=='2020-06-30']

# 연령대별 치료일수 비교
# 나이 중 숫자만 비교하기 위해 추출
total_0630['나이_int'] = total_0630['환자의 나이'].apply(lambda x : x[:-1]).astype('int')

## Age_Group 추가
def age(x):
    if x <= 10 : return '미성년자'
    elif x <= 30: return '청년층'
    elif x <= 50 : return '중년층'
    else: return '고령층'

total_0630['연령층'] = total_0630['나이_int'].apply(age)

# 컬럼 추가
total_0630['누적 확진자 수 비율'] = round(total_0630['누적 확진자 수']/(total_0630['누적 확진자 수'].sum()) * 100,2)
total_0630['누적 사망자 수 비율'] = round(total_0630['누적 사망자 수']/(total_0630['누적 사망자 수'].sum()) * 100,2)

total_0630.head()

# line charts : 연령별 누적 확진자 수 추이 그래프
fig = px.bar(covid_TimeAge, x='날짜', y='누적 확진자 수',
             color = "환자의 나이", color_discrete_sequence = px.colors.qualitative.Pastel)

fig.update_layout(title='<b>연령별 누적 확진자 수 추이</b>',
                  plot_bgcolor="#FFFFFF",yaxis_gridcolor = '#D5D5D5')

fig.show()

```python
# 데이터 불러오기 : covid_TimeAge.csv
covid_TimeAge = pd.read_csv('../data/TimeAge.csv')

# 컬럼명 변경 : 테이블 정의서 참고
covid_TimeAge.columns = ['날짜','시간','환자의 나이','누적 확진자 수','누적 사망자 수']

# 불필요한 컬럼 제거
covid_TimeAge.drop(['시간'], axis=1, inplace=True)

covid_TimeAge.head()

# 마지막 집계일(2020.06.30)만 추출
total_0630 = covid_TimeAge[covid_TimeAge['날짜']=='2020-06-30']

# 연령대별 치료일수 비교
# 나이 중 숫자만 비교하기 위해 추출
total_0630['나이_int'] = total_0630['환자의 나이'].apply(lambda x : x[:-1]).astype('int')

## Age_Group 추가
def age(x):
    if x <= 10 : return '미성년자'
```
```

```
    elif x <= 30: return '청년층'
    elif x <= 50 : return '중년층'
    else: return '고령층'

total_0630['연령층'] = total_0630['나이_int'].apply(age)

# 컬럼 추가
total_0630['누적 확진자 수 비율'] = round(total_0630['누적 확진자 수']/(total_0630['누적 확진자 수'].sum()) * 100,2)
total_0630['누적 사망자 수 비율'] = round(total_0630['누적 사망자 수']/(total_0630['누적 사망자 수'].sum()) * 100,2)

total_0630.head()

# line charts : 연령별 누적 확진자 수 추이 그래프
fig = px.bar(covid_TimeAge, x='날짜', y='누적 확진자 수',
             color = "환자의 나이", color_discrete_sequence = px.colors.qualitative.Pastel)

fig.update_layout(title='<b>연령별 누적 확진자 수 추이</b>',
                  plot_bgcolor="#FFFFFF",yaxis_gridcolor = '#D5D5D5')

fig.show()

# line charts :연령별 누적 확진자 수 추이 그래프
fig = px.bar(covid_TimeAge, x='날짜', y='누적 사망자 수',
             color = "환자의 나이", color_discrete_sequence = px.colors.qualitative.Pastel)

fig.update_layout(title='<b>연령별 누적 사망자 수 추이</b>',
                  plot_bgcolor="#FFFFFF", yaxis_gridcolor = '#D5D5D5')

fig.show()

# pie chart : 연령별 사망자 비중
fig = px.pie(total_0630, names='연령층' ,values='누적 사망자 수',
             title='<b>연령별 사망자 수 (2020.06.30)</b>',
             color_discrete_sequence = px.colors.colorbrewer.Set2)

fig.update_traces(textinfo='percent+label', textfont_size=13, hole=.3)

fig.show()


# 연령별 치사율 비교
# 치사율 컬럼 추가(소수점 둘째 자리까지만 표시)
covid_TimeAge['치사율'] = round(covid_TimeAge['누적 사망자 수']/covid_TimeAge['누적 확진자 수']*100, 2)

# 데이터 불러오기 : covid_PatientInfo.csv
covid_PatientInfo = pd.read_csv('../data/PatientInfo.csv')

# 컬럼명 변경 : 테이블 정의서 참고
covid_PatientInfo.columns = ['ID','성별','나이','국가','지방','도시',
                             '감염경로','감염시킨 사람의 ID','사람들과의 접촉 수',
                             '증상 발현 날짜','확진날짜','퇴원날짜','사망날짜','격리상태']

# 필요 컬럼만 추출하여 변수에 저장
covid_hospital=covid_PatientInfo[['나이','확진날짜','퇴원날짜']]

# 결측치 제거
covid_hospital = covid_hospital.dropna(axis=0)

# Age 열 추가
covid_hospital['Age'] = covid_hospital['나이'].apply(lambda x : x[:-1]).astype('int')

# 연령층 열 추가
def age(x):
    if x <= 10 : return '미성년자'
    elif x <= 30: return '청년층'
    elif x <= 50 : return '중년층'
    else: return '고령층'

covid_hospital['연령층'] = covid_hospital['Age'].apply(age)

covid_hospital.head()
나이 확진날짜  퇴원


# 결측치 제거
covid_hospital = covid_hospital.dropna(axis=0)

# 계산하기 위해 날짜 변환
covid_hospital['확진날짜']=pd.to_datetime(covid_hospital['확진날짜'])
covid_hospital['퇴원날짜']=pd.to_datetime(covid_hospital['퇴원날짜'])
```

```
# 치료일수 계산(치료일수 = 퇴원날짜 - 확진날짜)
covid_hospital['치료일수'] = covid_hospital['퇴원날짜'] - covid_hospital['확진날짜']

# 확진날짜 5월달 까지만
covid_hospital = covid_hospital[covid_hospital['확진날짜']<='2020-05-30']

# 치료일수 숫자로 변환
covid_hospital['치료일수']=round(pd.to_numeric(covid_hospital['치료일수'].dt.days, downcast='integer'),2)

# 시각화
# 연령층별 치료일수 평균 도출
age_group_period = covid_hospital[['치료일수','연령층']].groupby(by='연령층').mean()

# 연령순으로 변경
age_group_period = age_group_period.reindex(index=['미성년자','청년층','중년층','고령층'])

# 인덱스 정리
age_group_period =  age_group_period.reset_index()

# 연령별 평균 치료일수 그래프
fig = px.bar(age_group_period, x='연령층', y='치료일수',
             color='연령층', color_discrete_sequence = px.colors.colorbrewer.Set2)

fig.update_layout(title='<b>연령층별 평균 치료일수</b>', plot_bgcolor="#FFFFFF")

fig.show()
```

3. 유동인구와 코로나 감염과의 상관관계 분석

가설 검증 : 유동인구가 많은 지역에 코로나 감염자 수도 많을 것이다.

```
# 행정동 코드 파일 불러오기 : adstrd_master.csv
# 변수명 : loc
loc = pd.read_csv('../data/adstrd_master.csv')

# 컬럼명 변경
loc.columns = ['행정동 코드', '동명', '시', '시군구']

loc.head()

# 2019년 1~3월 유동인구 파일(KT) 불러오기 : fpopl_2019.csv
# 변수명 : pop2019_kt
pop2019_kt = pd.read_csv('../data/fpopl_2019.csv')

# 컬럼명 변경
pop2019_kt.columns = ['일자', '시간대', '성별', '나이대', '행정동 코드', '유동인구 수']

# 결과 확인
# pop2019_kt.head()

# 2019년 4, 5월 유동인구 파일(SKT) 불러오기 : FLT_SEOUL_04MONTH.csv, 'FLT_SEOUL_05MONTH.csv
skt_4 = pd.read_csv('../data/FLT_SEOUL_04MONTH.csv')
skt_5 = pd.read_csv('../data/FLT_SEOUL_05MONTH.csv')

# 4,5월 데이터 병합 : pop2019_skt
pop2019_skt = pd.concat([skt_4, skt_5])

# 컬럼명 변경
pop2019_skt.columns = ['일자', '시간대', '나이대', '성별', '시', '시군구', '유동인구 수']

# 결과 확인
# pop2019_skt.head()


# 행정동 코드 파일 불러오기 : adstrd_master.csv
# 변수명 : loc
loc = pd.read_csv('../data/adstrd_master.csv')

# 컬럼명 변경
loc.columns = ['행정동 코드', '동명', '시', '시군구']

loc.head()

# 2019년 1~3월 유동인구 파일(KT) 불러오기 : fpopl_2019.csv
# 변수명 : pop2019_kt
pop2019_kt = pd.read_csv('../data/fpopl_2019.csv')

# 컬럼명 변경
```

```
pop2019_kt.columns = ['일자', '시간대', '성별', '나이대', '행정동 코드', '유동인구 수']

# 결과 확인
# pop2019_kt.head()

# 2019년 4, 5월 유동인구 파일(SKT) 불러오기 : FLT_SEOUL_04MONTH.csv, 'FLT_SEOUL_05MONTH.csv
skt_4 = pd.read_csv('../data/FLT_SEOUL_04MONTH.csv')
skt_5 = pd.read_csv('../data/FLT_SEOUL_05MONTH.csv')

# 4,5월 데이터 병합 : pop2019_skt
pop2019_skt = pd.concat([skt_4, skt_5])

# 컬럼명 변경
pop2019_skt.columns = ['일자', '시간대', '나이대', '성별', '시', '시군구', '유동인구 수']

# 결과 확인
# pop2019_skt.head()

# 2019년 전연령 유동인구 데이터 전처리 (1~5월)

# 행정동 코드를 기준으로 2019 유동인구 데이터에 '동명', '시', '구' 열 추가 후 병합
pop2019_kt = pd.merge(pop2019_kt, loc, on='행정동 코드')

# '시' == 서울특별시 조건 충족 열 추출 : pop2019_seoul
pop2019_seoul = pop2019_kt[pop2019_kt['시']=='서울특별시']

# kt와 skt 유동인구 데이터 병합을 위해 필요한 열만 저장
pop2019_all_kt = pop2019_kt[['일자','시군구','유동인구 수']]
pop2019_all_skt = pop2019_skt[['일자','시군구','유동인구 수']]

# pop2019_kt, pop2019_skt 데이터를 행 병합 : pop2019
pop2019 = pd.concat([pop2019_all_kt,pop2019_all_skt])

pop2019.head()

# 구별로 전연령 유동인구 수 합산 : pop2019_all_gu
pop2019_all_gu = pop2019.groupby(pop2019.시군구)[['유동인구 수']].sum()

# 전연령 유동인구 수 기준으로 내림차순 정렬
pop2019_all_gu.sort_values('유동인구 수',ascending=False).head()
```

```
# 2019년 구별로 60대 이상 고령자 유동인구 수 추출

# pop2019_seoul에서 나이대가 60대 이상인 행 추출
# 데이터 프레임 생성 : pop60, pop65, pop70
pop60 = pop2019_seoul[pop2019_seoul['나이대']=='age_60']
pop65 = pop2019_seoul[pop2019_seoul['나이대']=='age_65']
pop70 = pop2019_seoul[pop2019_seoul['나이대']=='age_70']

# 데이터 병합 : pop2019_old_kt
pop2019_old_kt = pd.concat([pop60,pop65,pop70])

# pop2019_skt에서 나이대가 60대 이상인 행 추출
# 데이터 프레임 생성 : pop60, pop70
pop60 = pop2019_skt[pop2019_skt['나이대']==60]
pop70 = pop2019_skt[pop2019_skt['나이대']==70]

# 데이터 병합 : pop2019_old_skt
pop2019_old_skt = pd.concat([pop60,pop70])

# pop2019_old_kt, pop2019_old_skt 데이터 병합하기 위해 필요한 열만 추출
pop2019_old_kt2 = pop2019_old_kt[['일자','시군구','유동인구 수']]
pop2019_old_skt2 = pop2019_old_skt[['일자','시군구','유동인구 수']]

# pop2019_kt, pop2019_skt 데이터를 행 병합 : pop2019_old
pop2019_old = pd.concat([pop2019_old_kt2,pop2019_old_skt2])

# pop2019_old.head()

# 구별로 고령자 유동인구 수 합산 : pop2019_old_gu
pop2019_old_gu = pop2019_old.groupby(pop2019_old.시군구)[['유동인구 수']].sum()

# 유동인구 수 기준으로 내림차순 정렬
# pop2019_old_gu.sort_values('유동인구 수',ascending=False).head()

# 전연령 유동인구와 고령자 유동인구 데이터 열로 병합
# 인덱스 해제
pop2019_all_gu.reset_index(inplace=True)
```

```
pop2019_old_gu.reset_index(inplace=True)

# 컬럼명 변경
pop2019_all_gu.rename(columns = {'유동인구 수':'전연령_유동인구_2019'}, inplace=True)
pop2019_old_gu.rename(columns = {'유동인구 수':'고령_유동인구_2019'}, inplace=True)

# 구를 기준으로 열 병합 : pop2019__gu
pop2019_gu = pd.merge(pop2019_all_gu, pop2019_old_gu, on='시군구', left_index=True)

pop2019_gu.head()

# 2020년 전연령 유동인구 데이터 전처리 (1~5월)

# 행정동 코드를 기준으로 2020 유동인구 데이터에 '동명', '시', '구' 열 추가 후 재할당
pop2020_kt = pd.merge(pop2020_kt, loc, on='행정동 코드')

# '시'가 서울특별시인 곳만 추출  : pop2020_seoul
pop2020_seoul = pop2020_kt[pop2020_kt['시']=='서울특별시']

## kt와 skt 유동인구 데이터 병합을 위해 필요한 열만 저장
pop2020_all_kt = pop2020_seoul[['일자','시군구','유동인구 수']]
pop2020_all_skt = pop2020_skt[['일자','시군구','유동인구 수']]

# pop2020_kt, pop2020_skt 데이터를 행 병합 : pop2020
pop2020 = pd.concat([pop2020_all_kt,pop2020_all_skt])

# pop2020.head()

# 구별로 전연령 유동인구 수 합산 후 pop2020_all_gu 로 저장
pop2020_all_gu = pop2020.groupby(pop2020.시군구)[['유동인구 수']].sum()

# 전연령 유동인구 수 기준으로 내림차순 정렬
pop2020_all_gu.sort_values('유동인구 수',ascending=False).head()

# 2020년 구별로 60대 이상 고령자 유동인구 수 추출

# pop2020_seoul에서 나이대가 60대 이상인 행만 추출
# 데이터 프레임 생성 : pop60, pop65, pop70
pop60 = pop2020_seoul[pop2020_seoul['나이대']=='age_60']
pop65 = pop2020_seoul[pop2020_seoul['나이대']=='age_65']
pop70 = pop2020_seoul[pop2020_seoul['나이대']=='age_70']

# 데이터 병합 : pop2020_old_kt
pop2020_old_kt = pd.concat([pop60,pop65,pop70])

# pop2020_skt에서 나이대가 60대 이상인 행만 추출
pop60 = pop2020_skt[pop2020_skt['나이대']==60]
pop70 = pop2020_skt[pop2020_skt['나이대']==70]

# 데이터 병합 : pop2020_old_skt
pop2020_old_skt = pd.concat([pop60,pop70])

# pop2020_old_kt, pop2020_old_skt 데이터 병합하기 위해 필요한 열만 추출
pop2020_old_kt2 = pop2020_old_kt[['일자','시군구','유동인구 수']]
pop2020_old_skt2 = pop2020_old_skt[['일자','시군구','유동인구 수']]

# pop2020_kt, pop2020_skt 데이터를 행 병합 : pop2020_old
pop2020_old = pd.concat([pop2020_old_kt2,pop2020_old_skt2])

# pop2020_old.head()

# 구별로 고령자 유동인구 수 합산 : pop2020_old_gu
pop2020_old_gu = pop2020_old.groupby(pop2020_old.시군구)[['유동인구 수']].sum()

# 고령자 유동인구 수 기준으로 내림차순 정렬
# pop2020_old_gu.sort_values('유동인구 수',ascending=False).head()

# 전연령 유동인구와 고령자 유동인구 데이터 열로 병합
# 인덱스 해제
pop2020_all_gu.reset_index(inplace=True)
pop2020_old_gu.reset_index(inplace=True)

# 컬럼명 변경
pop2020_all_gu.rename(columns = {'유동인구 수':'전연령_유동인구_2020'}, inplace=True)
pop2020_old_gu.rename(columns = {'유동인구 수':'고령_유동인구_2020'}, inplace=True)

# 구를 기준으로 열 병합 후 pop2020_gu로 저장
pop2020_gu = pd.merge(pop2020_all_gu, pop2020_old_gu, on='시군구', left_index=True)

pop2020_gu.head()
```

```
# 사용할 데이터셋 : pop2019_gu , pop2020_gu
# pop2019_gu.head()

# 2019년
# 전연령 유동인구 수와 60대 이상 고령자 유동인구 수 상관관계 분석
sns.lmplot(x='고령_유동인구_2019', y='전연령_유동인구_2019',
           data=pop2019_gu, height=8)
plt.title("2019년 전연령 유동인구 수와 고령자 유동인구 수 상관관계", size = 15)
```

```
# 구체적인 상관계수 확인
np.corrcoef(pop2019_gu['고령_유동인구_2019'], pop2019_gu['전연령_유동인구_2019'])
# 결과 : 높은 상관관계
### 1.2. 2020년 1~5월 자치구별 전체 유동인구와 고령 유동인구 수 상관관계 분석
### 2020년
# 전연령 유동인구 수와 60대 이상 고령자 유동인구 수 상관관계 분석
sns.lmplot(x='고령_유동인구_2020', y='전연령_유동인구_2020',
           data=pop2020_gu, height=8)
plt.title("2020년 전연령 유동인구 수와 고령자 유동인구 수 상관관계", size = 15)
```

결과 분석

- 코로나 데이터에 결측치가 다수 존재. 확진자의 연령 구분이 어려움
- 전체 확진자 수를 분석하는 대신 **전체 유동인구와 60대 이상 고령 유동인구의 상관관계**를 분석
- 2019년, 2020년 1~5월 자치구별 유동인구 수를 전연령, 60대 이상 고령자로 나누어 둘의 상관관계를 분석
- 2019년의 **상관계수가 0.892 , 2020년의 상관계수가 0.896**로, 매우 높은 상관관계가 있음을 확인