서울 지하철 분석 및 시각화

- 데이터 분석 및 시각화 프로젝트

김 우 기

CONTENTS

1 목적

1 지하철 공공데이터의 컬럼 간 상관관계를 확인

2 노선/월/요일 별 지하철 승/하차 인원을 파악

3 파악한 데이터를 분석하고 시각화

4 분석한 데이터를 기반으로 다음달의 승/하차 인원을 예측

2 데이터 생성 – 생성 및 확인

서울 열린데이터 광장에서
"지하철호선별 역별 승하차 인원 정보" 데이터 활용

CARD_SUBWAY_MONTH_202201
CARD_SUBWAY_MONTH_202202
CARD_SUBWAY_MONTH_202203
CARD_SUBWAY_MONTH_202204
CARD_SUBWAY_MONTH_202205
CARD_SUBWAY_MONTH_202206
CARD_SUBWAY_MONTH_202207
CARD_SUBWAY_MONTH_202208

csv 파일 가져오기

```
df1=pd.read_csv("Metro/CARD_SUBWAY_MONTH_202201.csv")
df2=pd.read_csv("Metro/CARD_SUBWAY_MONTH_202202.csv")
df3=pd.read_csv("Metro/CARD_SUBWAY_MONTH_202203.csv")
df4=pd.read_csv("Metro/CARD_SUBWAY_MONTH_202204.csv")
df5=pd.read_csv("Metro/CARD_SUBWAY_MONTH_202205.csv")
df6=pd.read_csv("Metro/CARD_SUBWAY_MONTH_202206.csv")
df7=pd.read_csv("Metro/CARD_SUBWAY_MONTH_202207.csv")
df8=pd.read_csv("Metro/CARD_SUBWAY_MONTH_202208.csv")
```

데이터 확인

df1

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
20220101	3호선	수서	7370	7076	20220104	NaN
20220101	3호선	학여울	461	473	20220104	NaN
20220101	3호선	대청	3224	2903	20220104	NaN
20220101	3호선	일원	3321	2803	20220104	NaN
20220101	경원선	창동	1	0	20220104	NaN
...	...	...	...	...	...	...
20220131	우이신설선	솔밭공원	975	966	20220203	NaN
20220131	우이신설선	북한산우이	1641	1683	20220203	NaN
20220131	안산선	수리산	1333	1225	20220203	NaN
20220131	안산선	오이도	3753	3975	20220203	NaN
20220131	안산선	정왕	3878	3739	20220203	NaN

18521 rows × 6 columns

컬럼이 밀린 것을 확인

등록일자 컬럼 삭제

```
df1=df1.drop(["등록일자"], axis=1)
df2=df2.drop(["등록일자"], axis=1)
df3=df3.drop(["등록일자"], axis=1)
df4=df4.drop(["등록일자"], axis=1)
df5=df5.drop(["등록일자"], axis=1)
df6=df6.drop(["등록일자"], axis=1)
df7=df7.drop(["등록일자"], axis=1)
df8=df8.drop(["등록일자"], axis=1)
```

등록일자
컬럼 삭제하고 저장

Index 컬럼을 활용

df01

	Unnamed: 0	사용일자	노선명	역명	승차총승객수	하차총승객수
0	20220101	3호선	수서	7370	7076	20220104
1	20220101	3호선	학여울	461	473	20220104
2	20220101	3호선	대청	3224	2903	20220104
3	20220101	3호선	일원	3321	2803	20220104
4	20220101	경원선	창동	1	0	20220104
...	...	...	...	...	...	...
18516	20220131	우이신설선	솔밭공원	975	966	20220203
18517	20220131	우이신설선	북한산우이	1641	1683	20220203
18518	20220131	안산선	수리산	1333	1225	20220203
18519	20220131	안산선	오이도	3753	3975	20220203
18520	20220131	안산선	정왕	3878	3739	20220203

컬럼명 변경

```
df01.columns=['날짜', '노선', '역', '승차승객', '하차승객', '등록일']
df02.columns=['날짜', '노선', '역', '승차승객', '하차승객', '등록일']
df03.columns=['날짜', '노선', '역', '승차승객', '하차승객', '등록일']
df04.columns=['날짜', '노선', '역', '승차승객', '하차승객', '등록일']
df05.columns=['날짜', '노선', '역', '승차승객', '하차승객', '등록일']
df06.columns=['날짜', '노선', '역', '승차승객', '하차승객', '등록일']
df07.columns=['날짜', '노선', '역', '승차승객', '하차승객', '등록일']
df08.columns=['날짜', '노선', '역', '승차승객', '하차승객', '등록일']
```

컬럼 수정 및 삭제

컬럼 추가

```
df01.insert(5, "총 승객", df01["승차승객"]+df01["하차승객"])
df02.insert(5, "총 승객", df02["승차승객"]+df02["하차승객"])
df03.insert(5, "총 승객", df03["승차승객"]+df03["하차승객"])
df04.insert(5, "총 승객", df04["승차승객"]+df04["하차승객"])
df05.insert(5, "총 승객", df05["승차승객"]+df05["하차승객"])
df06.insert(5, "총 승객", df06["승차승객"]+df06["하차승객"])
df07.insert(5, "총 승객", df07["승차승객"]+df07["하차승객"])
df08.insert(5, "총 승객", df08["승차승객"]+df08["하차승객"])
```

```
df01.insert(6, "승차-하차", df01["승차승객"]-df01["하차승객"])
df02.insert(6, "승차-하차", df02["승차승객"]-df02["하차승객"])
df03.insert(6, "승차-하차", df03["승차승객"]-df03["하차승객"])
df04.insert(6, "승차-하차", df04["승차승객"]-df04["하차승객"])
df05.insert(6, "승차-하차", df05["승차승객"]-df05["하차승객"])
df06.insert(6, "승차-하차", df06["승차승객"]-df06["하차승객"])
df07.insert(6, "승차-하차", df07["승차승객"]-df07["하차승객"])
df08.insert(6, "승차-하차", df08["승차승객"]-df08["하차승객"])
```

df01

	날짜	노선	역	승차승객	하차승객	총 승객	승차-하차
0	20220101	3호선	수서	7370	7076	14446	294
1	20220101	3호선	학여울	461	473	934	-12
2	20220101	3호선	대청	3224	2903	6127	321
3	20220101	3호선	일원	3321	2803	6124	518
4	20220101	경원선	창동	1	0	1	1
...	...	...	...	...	...	...	...
18516	20220131	우이신설선	솔밭공원	975	966	1941	9
18517	20220131	우이신설선	북한산우이	1641	1683	3324	-42
18518	20220131	안산선	수리산	1333	1225	2558	108
18519	20220131	안산선	오이도	3753	3975	7728	-222
18520	20220131	안산선	정왕	3878	3739	7617	139

2 데이터 생성 – 날짜 형 변환

날짜 형 변환

- int -> object -> datetime

```
df01['날짜']= df01['날짜'].astype('str')
df02['날짜']= df02['날짜'].astype('str')
df03['날짜']= df03['날짜'].astype('str')
df04['날짜']= df04['날짜'].astype('str')
df05['날짜']= df05['날짜'].astype('str')
df06['날짜']= df06['날짜'].astype('str')
df07['날짜']= df07['날짜'].astype('str')
df08['날짜']= df08['날짜'].astype('str')
```

```
df01['날짜'] = pd.to_datetime(df01['날짜'])
df02['날짜'] = pd.to_datetime(df02['날짜'])
df03['날짜'] = pd.to_datetime(df03['날짜'])
df04['날짜'] = pd.to_datetime(df04['날짜'])
df05['날짜'] = pd.to_datetime(df05['날짜'])
df06['날짜'] = pd.to_datetime(df06['날짜'])
df07['날짜'] = pd.to_datetime(df07['날짜'])
df08['날짜'] = pd.to_datetime(df08['날짜'])
```

변환된 날짜 컬럼에서
월과 일을 추출

월/일 컬럼 추가

```
# df01['날짜'].dt.year # 연도 정보
df01['월']=df01['날짜'].dt.month # 월 정보
df01['일']=df01['날짜'].dt.day # 일 정보
df02['월']=df02['날짜'].dt.month # 월 정보
df02['일']=df02['날짜'].dt.day # 일 정보
df03['월']=df03['날짜'].dt.month # 월 정보
df03['일']=df03['날짜'].dt.day # 일 정보
df04['월']=df04['날짜'].dt.month # 월 정보
df04['일']=df04['날짜'].dt.day # 일 정보
df05['월']=df05['날짜'].dt.month # 월 정보
df05['일']=df05['날짜'].dt.day # 일 정보
df06['월']=df06['날짜'].dt.month # 월 정보
df06['일']=df06['날짜'].dt.day # 일 정보
df07['월']=df07['날짜'].dt.month # 월 정보
df07['일']=df07['날짜'].dt.day # 일 정보
df08['월']=df08['날짜'].dt.month # 월 정보
df08['일']=df08['날짜'].dt.day # 일 정보
```

Int형태의 날짜 컬럼을 문자열 형태로 변환 뒤 Datatime 형태로 변환

2 데이터 생성 – 요일 컬럼 추가

요일 추가

```
df01['weekday'] = df01['날짜'].dt.weekday
df02['weekday'] = df02['날짜'].dt.weekday
df03['weekday'] = df03['날짜'].dt.weekday
df04['weekday'] = df04['날짜'].dt.weekday
df05['weekday'] = df05['날짜'].dt.weekday
df06['weekday'] = df06['날짜'].dt.weekday
df07['weekday'] = df07['날짜'].dt.weekday
df08['weekday'] = df08['날짜'].dt.weekday
```

요일이 숫자 형태로 나타남

	날짜	노선	역	승차승객	하차승객	총 승객	승차-하차	월	일	weekday
0	2022-03-01	장항선	배방	593	698	1291	-105	3	1	1
1	2022-03-01	장항선	온양온천	2388	2517	4905	-129	3	1	1
2	2022-03-01	장항선	신창(순천향대)	1065	1164	2229	-99	3	1	1
3	2022-03-01	안산선	오이도	4789	4668	9457	121	3	1	1
4	2022-03-01	안산선	수리산	1892	1693	3585	199	3	1	1

weekday를 한글 날짜로 변환

- 0 = 월, 1 = 화, 2 = 수, 3 = 목, 4= 금, 5 = 토, 6 = 일
- 한글로 변환 위해 리스트 필요

요일 리스트 생성

```
weekday_list = ['월', '화', '수', '목', '금', '토', '일']
```

```
df01['요일'] = df01.apply(lambda x : weekday_list[x['weekday']], axis = 1)
df02['요일'] = df02.apply(lambda x : weekday_list[x['weekday']], axis = 1)
df03['요일'] = df03.apply(lambda x : weekday_list[x['weekday']], axis = 1)
df04['요일'] = df04.apply(lambda x : weekday_list[x['weekday']], axis = 1)
df05['요일'] = df05.apply(lambda x : weekday_list[x['weekday']], axis = 1)
df06['요일'] = df06.apply(lambda x : weekday_list[x['weekday']], axis = 1)
df07['요일'] = df07.apply(lambda x : weekday_list[x['weekday']], axis = 1)
df08['요일'] = df08.apply(lambda x : weekday_list[x['weekday']], axis = 1)
```

람다식 활용하여 요일 적용

df07

	날짜	노선	역	승차승객	하차승객	총 승객	승차-하차	월	일	요일
0	2022-07-01	2호선	성수	43013	47661	90674	-4648	7	1	금
1	2022-07-01	2호선	건대입구	39433	44202	83635	-4769	7	1	금
2	2022-07-01	2호선	구의(광진구청)	25932	24750	50682	1182	7	1	금
3	2022-07-01	2호선	강변(동서울터미널)	36241	35859	72100	382	7	1	금
4	2022-07-01	2호선	잠실나루	16254	15555	31809	699	7	1	금
...	...	...	...	...	...	...	...	...	...	...
18785	2022-07-31	2호선	시청	6851	5522	12373	1329	7	31	일
18786	2022-07-31	2호선	을지로입구	16256	16241	32497	15	7	31	일
18787	2022-07-31	2호선	을지로3가	8336	8359	16695	-23	7	31	일
18788	2022-07-31	분당선	복정	1	0	1	1	7	31	일
18789	2022-07-31	분당선	수원	5473	4521	9994	952	7	31	일

2 데이터 생성 – 데이터 합치기

2201~2208 합치기

```
df1=pd.read_csv("./2201.csv")
df2=pd.read_csv("./2202.csv")
df3=pd.read_csv("./2203.csv")
df4=pd.read_csv("./2204.csv")
df5=pd.read_csv("./2205.csv")
df6=pd.read_csv("./2206.csv")
df7=pd.read_csv("./2207.csv")
df8=pd.read_csv("./2208.csv")
```

```
result1 = pd.concat([df1,df2,df3,df4,df5,df6,df7,df8])
```

```
result1
```

변환된 데이터 저장 후 불러와서
하나로 합침

```
result1=result1.to_csv("./2022.csv", index=False)
```

```
df10=pd.read_csv("./2022.csv")
df10
```

	날짜	노선	역	승차승객	하차승객	총 승객	승차-하차	월	일	요일
0	2022-01-01	3호선	수서	7370	7076	14446	294	1	1	토
1	2022-01-01	3호선	학여울	461	473	934	-12	1	1	토
2	2022-01-01	3호선	대청	3224	2903	6127	321	1	1	토
3	2022-01-01	3호선	일원	3321	2803	6124	518	1	1	토
4	2022-01-01	경원선	창동	1	0	1	1	1	1	토
...	...	...	...	...	...	...	...	...	...	...
145789	2022-08-31	중앙선	국수	901	848	1749	53	8	31	수
145790	2022-08-31	중앙선	아신	639	632	1271	7	8	31	수
145791	2022-08-31	중앙선	오빈	321	317	638	4	8	31	수
145792	2022-08-31	중앙선	양평	2850	2848	5698	2	8	31	수
145793	2022-08-31	중앙선	용문	1754	1726	3480	28	8	31	수

2022년 1월~8월까지의 서울 지하철 데이터 생성 완료

3 데이터 분석 및 시각화 – 상관관계 찾기

불러온 데이터 확인

```
df9.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145794 entries, 0 to 145793
Data columns (total 10 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   날짜      145794 non-null  object
 1   노선      145794 non-null  object
 2   역       145794 non-null  object
 3   승차승객    145794 non-null  int64
 4   하차승객    145794 non-null  int64
 5   총 승객    145794 non-null  int64
 6   승차-하차   145794 non-null  int64
 7   월       145794 non-null  int64
 8   일       145794 non-null  int64
 9   요일      145794 non-null  object
dtypes: int64(6), object(4)
memory usage: 11.1+ MB
```

```
df9.isna().sum()
```

```
날짜      0
노선      0
역       0
승차승객    0
하차승객    0
총 승객    0
승차-하차   0
월       0
일       0
요일      0
dtype: int64
```

결측치 없음

```
df9.describe(include="object")
```

	날짜	노선	역	요일
count	145794	145794	145794	145794
unique	243	26	529	7
top	2022-07-14	5호선	서울역	월
freq	609	13608	1215	21010

object 컬럼

- 날짜 중 가장 많은 데이터가 있는 날은 2022년 7월 14일
- 노선 중 가장 많은 데이터가 있는 노선은 5호선
- 역 중 가장 많은 데이터가 있는 역은 서울역
- 요일 중 가장 많은 데이터가 있는 요일은 월요일

Object 형까지 요약 및 빈번한 데이터 확인

```
df9.describe()
```

	승차승객	하차승객	총 승객	승차-하차	월	일
count	145794.000000	145794.000000	145794.000000	145794.000000	145794.000000	145794.000000
mean	9826.457275	9791.574585	19618.031860	34.882691	4.541161	15.706442
std	9877.384268	10016.668445	19864.623821	1090.620867	2.298354	8.791725
min	1.000000	0.000000	1.000000	-20299.000000	1.000000	1.000000
25%	3333.000000	3223.000000	6582.000000	-206.000000	3.000000	8.000000
50%	7000.500000	6821.500000	13852.000000	63.000000	5.000000	16.000000
75%	12953.000000	12796.000000	25805.750000	369.000000	7.000000	23.000000
max	122543.000000	118237.000000	240780.000000	28118.000000	8.000000	31.000000

불러온 데이터 요약-int 형의 데이터만 확인

승차/ 하차/ 총 승객이 서로 상관관계가 있음

```
corr=df9.corr()
corr
```

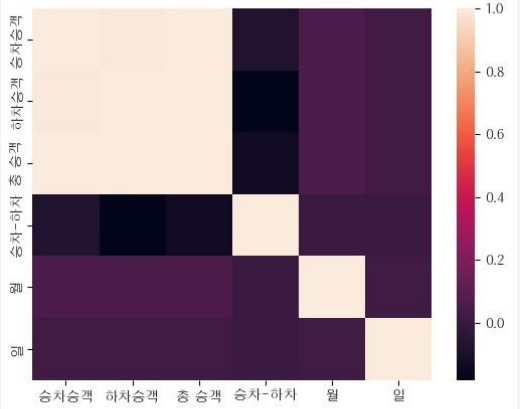
	승차승객	하차승객	총 승객	승차-하차	월	일
승차승객	1.000000	0.994087	0.998500	-0.073403	0.056150	0.023911
하차승객	0.994087	1.000000	0.998541	-0.181263	0.055057	0.023543
총 승객	0.998500	0.998541	1.000000	-0.127900	0.055682	0.023761
승차-하차	-0.073403	-0.181263	-0.127900	1.000000	0.002863	0.000331
월	0.056150	0.055057	0.055682	0.002863	1.000000	0.019195
일	0.023911	0.023543	0.023761	0.000331	0.019195	1.000000

결과 확인

- 승차승객과 하차승객, 총 승객
- 하차승객과 승차승객, 총 승객
- 총 승객과 승차승객, 하차승객
- 3개의 컬럼이 서로 상관관계를 갖고 있음

```
sns.heatmap(corr)
```

```
<AxesSubplot:>
```



상관관계 정리 ¶

- 승차승객, 하차승객, 총 승객은 서로 상관관계를 갖고 있음
- 그 외 데이터는 상관관계가 없음

히트맵을 통해 시각화

3 데이터 분석 및 시각화 – 승객 데이터

데이터의 표준편차 / 평균 / 중간 값

```
df9.std()
```

```
승차승객       9877.384268
하차승객      10016.668445
총 승객      19864.623821
승차-하차      1090.620867
월                2.298354
일                8.791725
dtype: float64
```

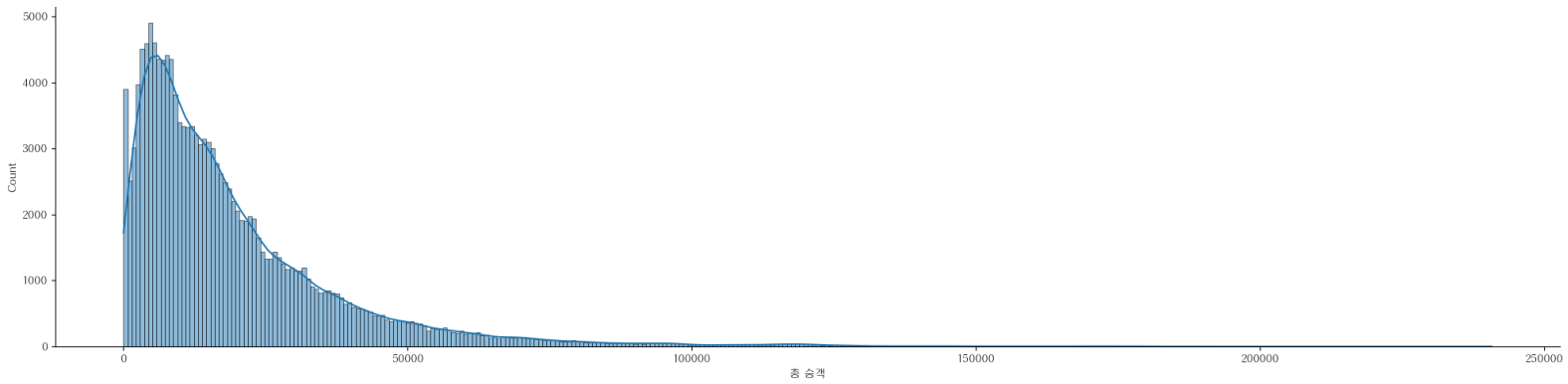
```
df9.mean()
```

```
승차승객       9826.457275
하차승객       9791.574585
총 승객      19618.031860
승차-하차        34.882691
월                4.541161
일               15.706442
dtype: float64
```

```
df9.median()
```

```
승차승객       7000.5
하차승객       6821.5
총 승객      13852.0
승차-하차        63.0
월                5.0
일               16.0
dtype: float64
```

```
sns.displot(data=df9, x="총 승객", kde=True, aspect=4)
```



총 승객 수 50,000 이하로
밀집되어 있고,
총 승객 수 100,000 이하가 대부분

250,000명에 근접한 수치도 있음

3 데이터 분석 및 시각화 – 노선

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.font_manager as fm
from wordcloud import WordCloud
from collections import Counter
from konlpy.tag import Okt
from PIL import Image
```

노선 별 분석을 위한 import

```
plt.rcParams['font.family']='HYGPRM'
```

Seaborn 그래프에서
한글 폰트 깨짐 방지

```
df9.노선.unique()
```

```
array(['3호선', '경원선', '1호선', '2호선', '경의선', '분당선', '중앙선', '장항선', '일산선',
       '우이신설선', '안산선', '수인선', '과천선', '공항철도 1호선', '경춘선', '경인선', '경부선',
       '경강선', '9호선2~3단계', '9호선', '8호선', '7호선', '6호선', '5호선', '4호선', '신림선'],
      dtype=object)
```

데이터에 있는 노선의 종류 확인

※ 참고사항
- 총 26개 노선
- 1~4월은 25개 노선, 5~8월은 26개 노선
- why? 신림선 2022년 5월 28일 개통

3 데이터 분석 및 시각화 – 노선

```
df9[["노선", "승차승객", "하차승객", "총 승객"]].groupby(["노선"]).mean().sort_values(by="총 승객", ascending=False)
```

```
df9[["노선", "승차승객", "하차승객", "총 승객"]].groupby(["노선"]).sum().sort_values(by="총 승객", ascending=False)
```

노선	승차승객	하차승객	총 승객
2호선	23320.706337	23624.886584	46945.592922
1호선	19378.279835	18952.500000	38330.779835
4호선	15882.019943	16064.927509	31946.947452
3호선	13119.977854	13093.300135	26213.277988
경인선	12148.306379	11868.020370	24016.326749
7호선	11809.916101	11628.988533	23438.904634
과천선	11153.318416	11045.965021	22199.283436
경부선	10015.789490	9949.541522	19965.331012
5호선	9731.628160	9676.739638	19408.367798
9호선	9467.898436	9572.574156	19040.472593
분당선	9199.009096	9496.680551	18695.689647
일산선	9194.705386	8854.735531	18049.440916
8호선	8923.063329	9002.865341	17925.928669
안산선	8088.442229	8059.795188	16148.237417
6호선	7509.098823	7407.034430	14916.133253
9호선2~3단계	6430.965812	6371.349794	12802.315606
공항철도 1호선	6521.818048	5998.872722	12520.690770
경원선	5989.060006	5828.779370	11817.839376
중앙선	3888.669018	3776.566725	7665.235744
경의선	3676.719592	3603.047022	7279.766614
우이신설선	3037.325419	2920.783159	5958.108579
신림선	2858.329545	2894.136364	5752.465909
수인선	2895.164609	2849.944902	5745.109511
경강선	2494.813692	2409.994014	4904.807707
장항선	1946.901822	1855.552616	3802.454439
경춘선	1849.485813	1766.315573	3615.801386

총 승객 평균 값

노선	승차승객	하차승객	총 승객
2호선	283346582	287042372	570388954
5호선	132427996	131681073	264109069
7호선	123590772	121697365	245288137
3호선	107229579	107011542	214241121
4호선	100342602	101498212	201840814
경부선	94919637	94291805	189211442
분당선	76857721	79344766	156202487
6호선	68918509	67981762	136900271
경인선	59040769	57678579	116719348
9호선	57517483	58153388	115670871
1호선	47089220	46054575	93143795
경원선	42618151	41477594	84095745
8호선	39029479	39378533	78408012
안산선	25551389	25460893	51012282
경의선	23457471	22987440	46444911
일산선	22876427	22030582	44907009
과천선	21682051	21473356	43155407
공항철도 1호선	22187225	20408165	42595390
9호선2~3단계	20315421	20127094	40442515
중앙선	19843878	19271820	39115698
수인선	12663450	12465659	25129109
우이신설선	9594911	9226754	18821665
경춘선	8539076	8155079	16694155
경강선	6668637	6441914	13110551
장항선	3311680	3156295	6467975
신림선	3018396	3056208	6074604

총 승객 합계

결과확인

총 승객 평균 정보

- 전체 이용자 수와 다른 부분이 있다
- 평균 총 승객은 2호선이 가장 많고, 경춘선이 가장 적다
- 2호선, 1호선, 4호선, 3호선, 경인선 순으로 많다
- 경춘선, 장항선, 경강선, 수인선, 신림선 순으로 적다

총 승객 합 정보

- 2호선, 5호선, 7호선, 3호선, 4호선 순으로 이용자가 많다
- 신림선, 장항선, 경강선, 경춘선, 우이신설선 순으로 이용자가 적다
- 2호선의 이용자가 압도적으로 많다

참고사항

- 신림선의 승객의 합이 가장 적은 것은 최근 개통의 영향

3 데이터 분석 및 시각화 – 노선

```
df9.노선.value_counts().sort_values(ascending=False)
```

```
5호선          13608
2호선          12150
7호선          10465
경부선          9477
6호선           9178
분당선          8355
3호선           8173
경원선          7116
경의선          6380
4호선           6318
9호선           6075
중앙선          5103
경인선          4860
경춘선          4617
8호선           4374
수인선          4374
공항철도 1호선     3402
9호선2~3단계      3159
안산선          3159
우이신설선        3159
경강선          2673
일산선          2488
1호선           2430
과천선          1944
장항선          1701
신림선          1056
Name: 노선, dtype: int64
```

노선 카운트 ¶

- 1호선, 4호선의 총 승객 대비 카운트 수가 적다
- 결과를 보면 1호선은 다른 노선에 비해서도 등장한 수가 적다
- 노선에 속한 역의 수가 많을수록 카운트도 많아진다
- 1호선에 속한 노선의 수는 적은 것으로 추정된다
- 다만, 1호선에 속한 역을 이용하는 사람은 전반적으로 높은 것으로 추정된다

데이터에서 노선 별 등장하는 횟수

노선별 역의 수

```
line_name = np.sort(df9.노선.unique())
line_code = pd.DataFrame(list(enumerate(line_name)), columns=['노선코드','노선'])
line_code
```

```
s = df9[['노선', '역']].drop_duplicates().노선.value_counts()
line_code['역수'] = line_code.노선.map(s)
line_code
```

노선 별 역의 개수

	노선코드	노선	역수
0	0	1호선	10
1	1	2호선	50
2	2	3호선	34
3	3	4호선	26
4	4	5호선	56
5	5	6호선	39
6	6	7호선	51
7	7	8호선	18
8	8	9호선	25
9	9	9호선2~3단계	13
10	10	경강선	11
11	11	경부선	39
12	12	경원선	30
13	13	경의선	29
14	14	경인선	20
15	15	경춘선	19
16	16	공항철도 1호선	14
17	17	과천선	8
18	18	분당선	35
19	19	수인선	18
20	20	신림선	11
21	21	안산선	13
22	22	우이신설선	13
23	23	일산선	11
24	24	장항선	7
25	25	중앙선	21

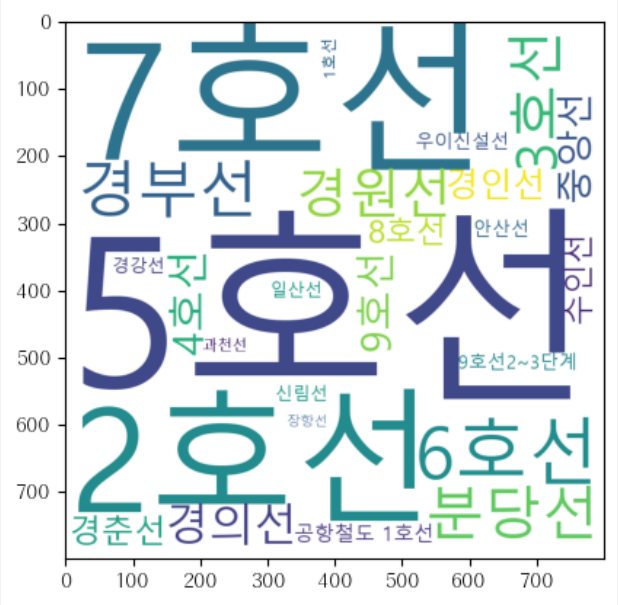
1호선이 역의 수가 가장 적고,
5호선이 역의 수가 가장 많다

5-7-2 호선 순으로 많고,
1-경강선-일산선 순으로 적다

3 데이터 분석 및 시각화 – 노선

```
wc=line_code.set_index("노선").to_dict()["역수"]

wordcloud = WordCloud(font_path='malgun', width=400, height=400, scale=2.0, max_font_size=250, background_color="white")
gen =wordcloud.generate_from_frequencies(wc)
plt.figure()
plt.imshow(gen)
```



역의 수를 토대로 만든 노선 워드 클라우드

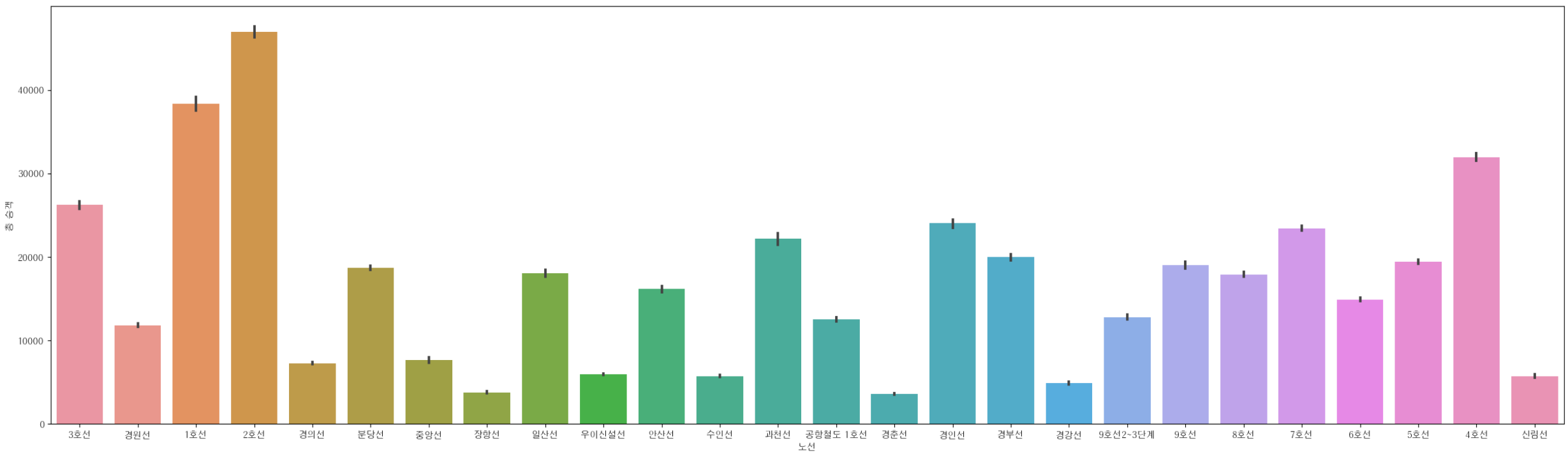
월 별/ 노선 별 총 승객 피벗 테이블

```
df9a=df9.pivot_table(index="월", columns="노선", values="총 승객")
df9a
```

노선 / 월	1호선	2호선	3호선	4호선	5호선	6호선	7호선	8호선	9호선	9호선2~3단계	...	공항철도 1호선	과천선	분당선	수인선	신림선	안산선	우이신설선
1	35196.625806	43653.232903	24383.783862	29437.203474	18055.775346	13663.841474	20945.650215	16855.062724	17813.563871	11662.337469	...	11004.529954	19960.165323	17423.384255	5081.691756	NaN	14464.982630	5468.786600
2	32398.550000	40774.674286	22898.817021	27653.245879	16962.439413	12963.789174	19951.843028	15775.049603	16610.682857	11015.024725	...	10342.211735	18597.991071	16272.207900	4790.833333	NaN	13578.607143	5277.629121
3	33276.367742	41767.422581	23151.466859	28874.148883	17405.872696	13553.235495	21355.996283	16236.259857	16689.815484	10955.218362	...	10438.414747	19407.415323	16815.561502	5327.740143	NaN	15026.724566	5512.449132
4	39388.526667	48533.842000	26912.061204	33158.924359	20305.444048	15544.340407	24920.967033	18887.494444	19853.649333	12913.566667	...	12502.252381	23723.575000	19520.956268	6262.320370	NaN	17443.133333	6242.748718
5	42815.854839	51541.072903	29270.865201	35861.838710	21311.559332	16655.410562	26390.007628	19506.910394	20800.927742	13771.838710	...	13884.562212	25646.935484	20700.681989	6708.229391	6418.000000	18588.818859	6670.260546
6	41943.436667	50286.223333	28387.732938	34554.153846	20791.584524	16154.348940	25492.059335	19131.672222	20492.433333	14060.756410	...	13920.595238	24134.850000	20022.055233	6301.644444	5540.863636	17508.084615	6365.220513
7	41221.409677	50325.717419	27736.885687	33493.224566	20369.753456	15612.112436	24783.029839	18747.173835	20313.212903	14215.508685	...	14070.096774	23636.975806	19629.193639	5831.430108	5792.123167	16597.521092	6187.972705
8	39982.038710	48244.368387	26710.286538	32250.519851	19901.368664	15038.180426	23861.742944	18129.569892	19587.423226	13695.493797	...	13836.605991	22249.455645	19013.088263	5599.274194	5831.709677	15814.990074	5896.263027

3 데이터 분석 및 시각화 – 노선

```
plt.subplots(figsize=(30, 8))
sns.barplot(data=df9, x="노선", y="총 승객")
plt.show()
```



노선 별 총 승객 막대 그래프

3 데이터 분석 및 시각화 – 날짜

```
df9[["월", "승차승객", "하차승객", "총 승객"]].groupby(["월"]).mean().sort_values(by="총 승객", ascending=False)
```

월	승차승객	하차승객	총 승객
5	10997.577801	10961.165117	21958.742917
6	10529.076407	10490.024974	21019.101381
4	10321.086737	10282.937132	20604.023870
7	10301.967110	10263.144119	20565.111229
8	9926.206474	9889.340113	19815.546587
1	9060.095135	9028.133794	18088.228929
3	8851.885688	8822.759463	17674.645151
2	8517.294818	8489.189983	17006.484801

월별 총 승객의 평균치와 합

```
df9[["월", "승차승객", "하차승객", "총 승객"]].groupby(["월"]).sum().sort_values(by="총 승객", ascending=False)
```

월	승차승객	하차승객	총 승객
5	203411199	202737710	406148909
7	193573962	192844478	386418440
6	191408080	190698164	382106244
8	186434010	185741586	372175596
4	184200435	183519579	367720014
1	167802022	167210066	335012088
3	163467773	162929899	326397672
2	142341031	141871343	284212374

결과확인

총 승객 평균 정보

- 5월이 가장 많고, 2월이 가장 적다

총 승객 합 정보

- 5월이 가장 많고, 2월이 가장 적다

분석

- 상대적으로 춥고, 더운 겨울과 여름보다 봄에 승객이 많다
- 한 달의 기간이 긴 5월과 7월, 8월의 승객 수가 높다
- 1월은 겨울이라는 점과 설연휴가 있다는 점이 영향을 끼쳤을 것이다
- 2월은 28일까지라는 점이 승객의 수에 영향을 끼쳤을 것이다
- 2월과 3월이 적은 것은 코로나 확진자 수가 많이 나왔던 기간과 겹친다

일	승차승객	하차승객	총 승객
25	10983.816458	10948.149375	21931.965833
11	10869.529694	10835.478850	21705.008543
18	10862.120000	10827.255417	21689.375417
4	10805.513322	10769.104913	21574.618235
21	10316.330623	10280.807171	20597.137794
22	10279.676747	10243.550365	20523.227112
14	10258.160266	10223.784513	20481.944779
28	10258.911153	10221.908448	20480.819600
8	10142.739221	10109.060821	20251.800042
29	10127.738779	10088.451674	20216.190454
7	10070.047460	10032.953164	20103.000624
10	10053.586739	10019.287531	20072.874270
24	9995.590663	9961.635890	19957.226553
17	9965.734111	9930.658887	19896.392999
26	9898.355996	9863.378102	19761.734098
12	9875.796289	9840.831109	19716.627398
19	9846.025005	9810.019796	19656.044801
27	9836.211426	9802.196831	19638.408257
16	9797.113565	9762.873724	19559.987289
20	9742.084601	9706.849344	19448.933945
3	9728.487602	9694.476766	19422.964368
23	9685.546003	9650.644255	19336.190258
15	9666.395412	9631.097393	19297.492805
30	9508.247084	9472.920733	18981.167817
13	9388.763356	9357.767738	18746.531093
9	9147.756982	9113.840559	18261.597541
5	9085.288991	9046.123436	18131.412427
2	8901.124844	8868.698624	17769.823468
31	8850.120306	8818.563642	17668.683948
6	8644.205919	8611.153397	17255.359316
1	7654.610474	7621.399124	15276.009597

일	승차승객	하차승객	총 승객
25	52722319	52551117	105273436
11	52162873	51999463	104162336
18	52138176	51970826	104109002
4	51909686	51734780	103644466
21	49487438	49317032	98804470
14	49413558	49247970	98661528
28	49304327	49126492	98430819
22	49291050	49117824	98408874
8	48695291	48533601	97228892
7	48376508	48198307	96574815
10	48217002	48052503	96269505
24	47958844	47795929	95754773
17	47825558	47657232	95482790
26	47462617	47294898	94757515
12	47364319	47196626	94560945
19	47251074	47078285	94329359
27	47174470	47011336	94185806
16	47016348	46852031	93868379
20	46752264	46583170	93335434
3	46687012	46523794	93210806
23	46529363	46361695	92891058
15	46350366	46181112	92531478
13	44990954	44842423	89833377
9	43890938	43728207	87619145
5	43573046	43385208	86958254
2	42707597	42552016	85259613
29	42647908	42482470	85130378
6	41474900	41316314	82791214
30	39944146	39795740	79739886
1	36688548	36529366	73217914
31	26630012	26535058	53165070

결과확인

총 승객 평균 정보

- 25일이 가장 많고, 1일이 가장 적다

총 승객 합 정보

- 25일이 가장 많고, 31일이 가장 적다

분석

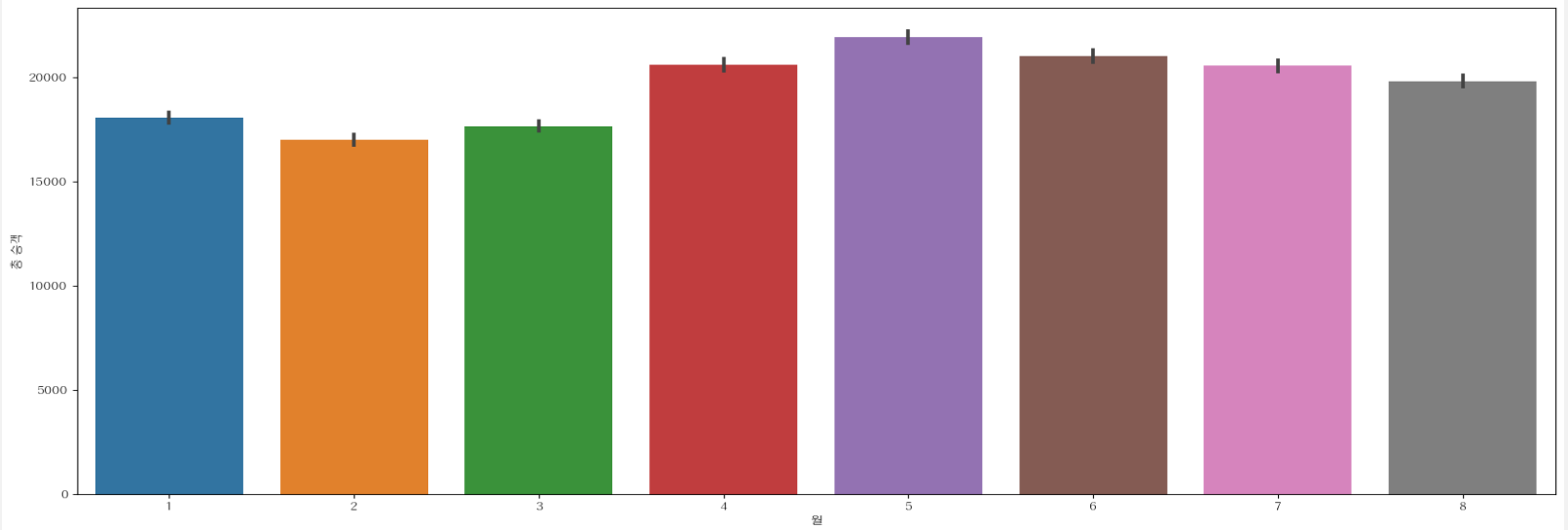
- 29일, 30일, 31일이 적은 것은 29일과 30일, 31일이 없는 달이 있어서다
- 휴일이 많은 날일수록 승객 수가 적다
- 매월 주중에 포함되는 날이 많을수록 승객 수가 많다
- 1일은 신정과 설연휴, 3.1절, 지방선거 등의 휴무일이 많아서 승객 수가 적다

일별 총 승객의 평균치와 합

3 데이터 분석 및 시각화 – 날짜

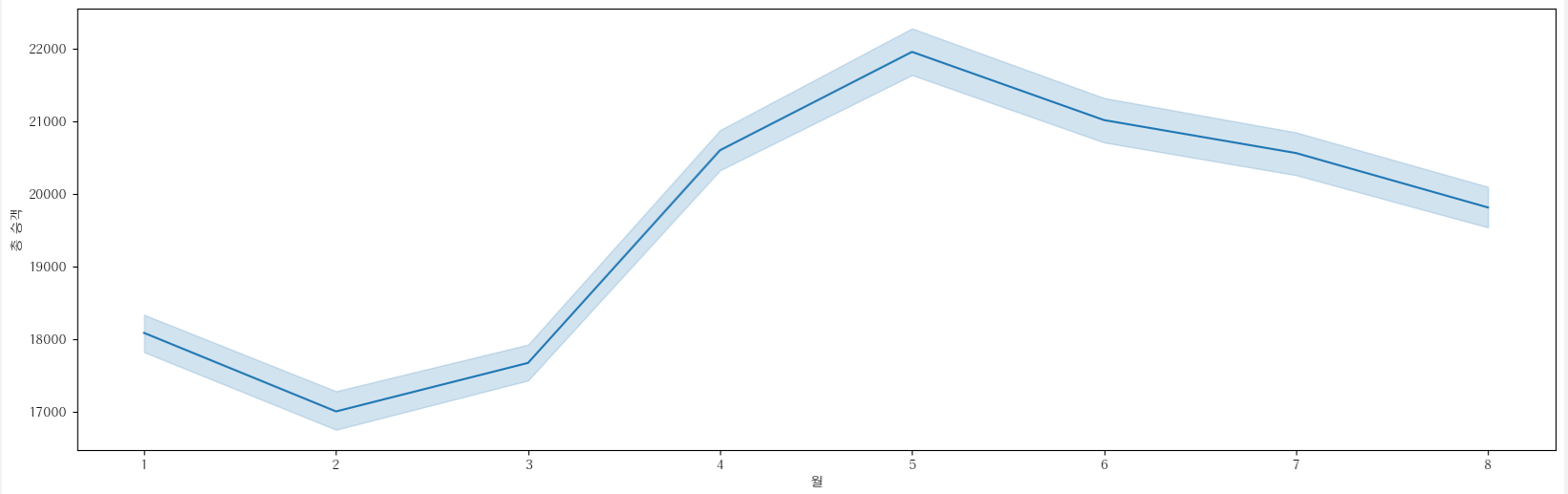
```
plt.subplots(figsize=(20, 8))
sns.barplot(data=df9, x="월", y="총 승객")
plt.show()
```

월별 총 승객 막대 그래프




```
plt.subplots(figsize=(20, 6))
sns.lineplot(data=df9, x="월", y="총 승객")
plt.show()
```

월별 총 승객 선 그래프



3 데이터 분석 및 시각화 – 요일

```
df9[["요일", "승차승객", "하차승객", "총 승객"]].groupby(["요일"]).mean().sort_values(by="총 승객", ascending=False)
```

요일	승차승객	하차승객	총 승객
금	11615.964436	11578.407808	23194.372245
목	11093.641590	11059.454127	22153.095717
화	10865.970183	10832.206525	21698.176709
수	10845.701881	10812.150012	21657.851892
월	10469.655878	10436.874964	20906.530842
토	8031.146395	7992.965282	16024.111677
일	5942.629578	5908.417159	11851.046738

```
df9[["요일", "승차승객", "하차승객", "총 승객"]].groupby(["요일"]).sum().sort_values(by="총 승객", ascending=False)
```

요일	승차승객	하차승객	총 승객
금	237128298	236361617	473489915
화	228131044	227422176	455553220
수	227813968	227109211	454923179
목	226354663	225657102	452011765
월	219967470	219278743	439246213
토	168638012	167836285	336474297
일	124605057	123887691	248492748

결과 확인

총 승객 평균 정보

- 주중의 승객 수가 많고, 주말의 승객 수가 적다
- 금요일 승객이 가장 많고, 일요일에 가장 적다

총 승객 합 정보

- 주중의 승객 수가 많고, 주말의 승객 수가 적다
- 금요일 승객이 가장 많고, 일요일에 가장 적다

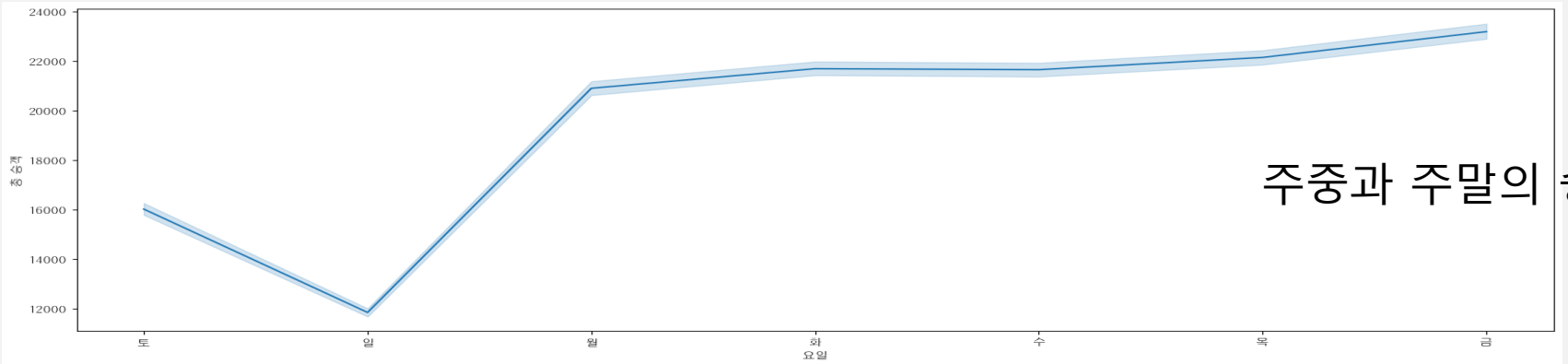
차이점

- 화, 수, 목은 평균치와 합에 따라 순서가 바뀌기도 한다
- 금요일이 승객이 가장 많고, 월, 토, 일 순으로 적은 것은 평균과 합이 같다

주중일수록 승객이 많고,
주말일수록 승객이 적다

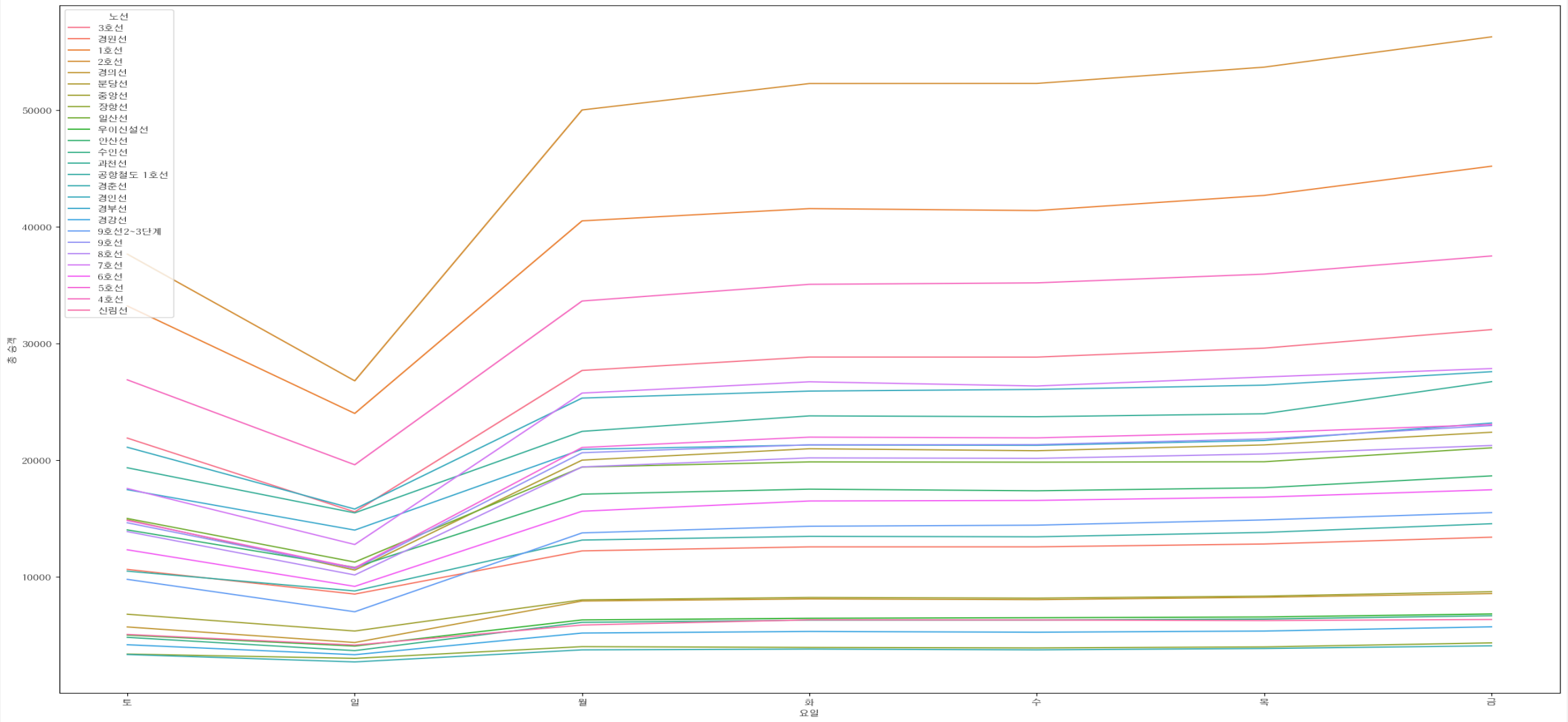
요일별 노선 총 승객 그래프

```
fig=plt.subplots(figsize=(26,15))
sns.lineplot(x=df9["요일"], y=df9["총 승객"], hue=df9["노선"], errorbar=None)
```



주중과 주말의 승객 차이를 명확하게 보여줌

3 데이터 분석 및 시각화 – 요일

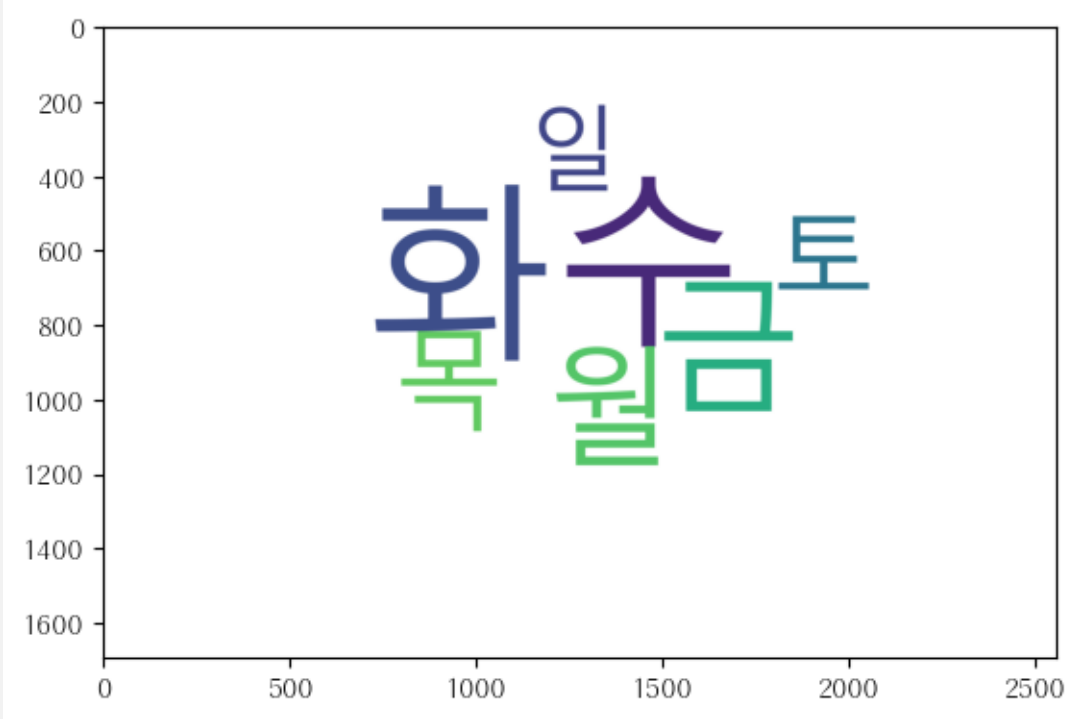


3 데이터 분석 및 시각화 – 요일

```
week_name = np.sort(df9.요일.unique())
week_code = pd.DataFrame(list(enumerate(week_name)), columns=['요일코드','요일'])
week_code
```

```
s = df9[['요일', '총 승객']].drop_duplicates().요일.value_counts()
week_code['총 승객'] = week_code.요일.map(s)
week_code
```

	요일코드	요일	총 승객
0	0	금	16623
1	1	목	16482
2	2	수	16775
3	3	월	16614
4	4	일	14138
5	5	토	15522
6	6	화	16753



요일 별 총 승객을 워드 클라우드로 표현

```
wordcloud = WordCloud(font_path='malgun', width=400, height=400, scale=2.0, max_font_size=250, background_color="white", mask=imgArray)
gen =wordcloud.generate_from_frequencies(wc)
plt.figure()
plt.imshow(gen)
```

3 데이터 분석 및 시각화 – 역

총 승객 평균 상위

역	승차승객	하차승객	총 승객
강남	71148.094650	69960.296296	141108.390947
구로디지털단지	49118.683128	48950.687243	98069.370370
삼성(무역센터)	44035.452675	44269.156379	88304.609053
역삼	40589.000000	45437.427984	86026.427984
서울대입구(관악구청)	41717.621399	40957.534979	82675.156379
잠실(송파구청)	39116.341564	39618.804527	78735.146091
신림	39530.327434	38372.513274	77902.840708
영등포	36468.798354	37382.646091	73851.444444
용산	33998.316872	34400.786008	68399.102881
을지로입구	32741.152263	33421.448560	66162.600823
양재(서초구청)	31132.864198	33900.646091	65033.510288
성수	30956.386831	33938.547325	64894.934156
혜화	29754.695473	30051.238683	59805.934156
수유(강북구청)	29766.028807	29732.213992	59498.242798
부천	29428.555556	29561.292181	58989.847737
종각	29246.827160	28284.251029	57531.078189
압구정	27530.407407	29424.864198	56955.271605
부평	27193.707819	29044.423868	56238.131687
사당	28010.199588	27992.176955	56002.376543
선릉	27708.485597	27883.442387	55591.927984

총 승객 합 상위

역	승차승객	하차승객	총 승객
잠실(송파구청)	19010542	19254739	38265281
강남	17288987	17000352	34289339
고속터미널	16712491	16651220	33363711
홍대입구	16095209	16956884	33052093
서울역	16402326	16425681	32828007
사당	13612957	13604198	27217155
선릉	13466324	13551353	27017677
신림	13400781	13008282	26409063
가산디지털단지	11904767	12446347	24351114
구로디지털단지	11935840	11895017	23830857
여의도	11127749	11207624	22335373
건대입구	10753431	11468587	22222018
신도림	11093283	11118199	22211482
삼성(무역센터)	10700615	10757405	21458020
역삼	9863127	11041295	20904422
종로3가	10421201	10313270	20734471
수원	10145310	10319837	20465147
서울대입구(관악구청)	10137382	9952681	20090063
합정	9808428	10224528	20032956
교대(법원.검찰청)	9533977	9385017	18918994

결과확인

총 승객 평균 정보

- 상위 역에는 2호선에 속해있는 역들이 많다
- 하위 역에는 7호선에 속해있는 역들이 많다
- 하위 9개 역의 총 승객 평균은 1점대이다
- 하위 9개 역은 하차 승객이 0이다

총 승객 합 정보

- 상위 역에는 2호선에 속해있는 역들이 많다
- 하위 역에는 7호선에 속해있는 역들이 많다
- 하위 9개 역의 총 승객 수는 100명이 안 된다
- 하위 9개 역은 하차 승객이 0이다

참고사항

- 총 승객 하위 역 중 하차 승객이 0인 역 중 일부는 환승역이어서 다른 노선으로 집계되기도 한다
- 7호선 일부 역들은 인천교통공사로 이관되어 서울시 데이터에 잡히지 않는다

총 승객 평균 하위

역	승차승객	하차승객	총 승객
원덕	377.411523	360.592593	738.004115
세종대왕릉	317.711934	317.506173	635.218107
상천	320.448560	292.259259	612.707819
아목	341.711934	257.300412	599.012346
오빈	293.740741	288.074074	581.814815
신원	222.123457	186.345679	408.469136
굴봉산	171.386831	165.781893	337.168724
백양리	164.967078	147.584362	312.551440
달월	91.839506	64.913580	156.753086
지평	65.917695	71.152263	137.069959
임진강	31.308642	24.855967	56.164609
삼산체육관	1.333333	0.000000	1.333333
신중동	1.333333	0.000000	1.333333
부천시청	1.323529	0.000000	1.323529
상동	1.272727	0.000000	1.272727
굴포천	1.206897	0.000000	1.206897
부평구청	1.185185	0.000000	1.185185
춘의	1.178571	0.000000	1.178571
부천종합운동장	1.111111	0.000000	1.111111
까치울	1.100000	0.000000	1.100000

총 승객 합 하위

역	승차승객	하차승객	총 승객
원덕	91711	87624	179335
세종대왕릉	77204	77154	154358
상천	77869	71019	148888
아목	83036	62524	145560
오빈	71379	70002	141381
신원	53976	45282	99258
굴봉산	41647	40285	81932
백양리	40087	35863	75950
달월	22317	15774	38091
지평	16018	17290	33308
임진강	7608	6040	13648
상동	56	0	56
부천시청	45	0	45
까치울	44	0	44
신중동	44	0	44
굴포천	35	0	35
춘의	33	0	33
부평구청	32	0	32
삼산체육관	20	0	20
부천종합운동장	10	0	10

역별 카운트

```
df9.역.value_counts().sort_values(ascending=False)
```

```
서울역          1215
공덕             972
김포공항         733
왕십리(성동구청)    729
종로3가          729
                ...
굴포천            29
춘의              28
부평구청          27
삼산체육관        15
부천종합운동장     9
Name: 역, Length: 529, dtype: int64
```

```
station_cross = df9[['역','노선']].drop_duplicates().sort_values('역').reset_index(drop=True)
station_cross
```

	역	노선
0	4.19민주묘지	우이신설선
1	가능	경원선
2	가락시장	8호선
3	가락시장	3호선
4	가산디지털단지	7호선
...	...	...
616	회룡	경원선
617	회현(남대문시장)	4호선
618	효창공원앞	6호선
619	효창공원앞	경의선
620	흑석(중앙대입구)	9호선

환승역 확인 및 개수

```
s = station_cross.역.value_counts()
s[s>2]
```

```
서울역          5
공덕            4
김포공항        4
홍대입구        3
디지털미디어시티    3
동대문역사문화공원(DDP)  3
왕십리(성동구청)    3
종로3가         3
신설동          3
고속터미널      3
Name: 역, dtype: int64
```

빈도 수 많을수록
환승역 개수 많다는 것 확인

빈도 수가 많을수록 환승역이 많을 것으로 판단

3 데이터 분석 및 시각화 – 역

역의 개수가 많아서 총 승객이 많은 25개의 역과
해당 역이 속한 노선 중 상위 5개 노선 한정

```
df9.groupby(["역"]).sum().sort_values(by="총 승객", ascending=False)[["총 승객"]].head(25)
```

```
top25_station=df9.groupby(["역"]).sum().sort_values(by="총 승객", ascending=False)[["총 승객"]].head(25).index
top25_station
```

```
Index(['잠실(송파구청)', '강남', '고속터미널', '홍대입구', '서울역', '사당', '선릉', '신림', '가산디지털단지',
       '구로디지털단지', '여의도', '건대입구', '신도림', '삼성(무역센터)', '역삼', '종로3가', '수원',
       '서울대입구(관악구청)', '합정', '교대(법원.검찰청)', '시청', '영등포', '노원', '용산',
       '청량리(서울시립대입구)'],
      dtype='object', name='역')
```

상위 25개 역의 리스트

상위 역 25개가 속해있는 노선

```
top25.노선.unique()
```

```
array(['1호선', '2호선', '분당선', '공항철도 1호선', '경의선', '경원선', '경부선', '9호선', '8호선',
       '7호선', '6호선', '5호선', '4호선', '3호선', '신림선'], dtype=object)
```

상위 25개 역이 속한 노선의 리스트

역	총 승객
잠실(송파구청)	38265281
강남	34289339
고속터미널	33363711
홍대입구	33052093
서울역	32828007
사당	27217155
선릉	27017677
신림	26409063
가산디지털단지	24351114
구로디지털단지	23830857
여의도	22335373
건대입구	22222018
신도림	22211482
삼성(무역센터)	21458020
역삼	20904422
종로3가	20734471
수원	20465147
서울대입구(관악구청)	20090063
합정	20032956
교대(법원.검찰청)	18918994
시청	18007884
영등포	17945901
노원	17288354
용산	16620982
청량리(서울시립대입구)	16081331

노선	총 승객
2호선	316925409
경부선	63131791
1호선	47405460
7호선	37742970
3호선	29010952
4호선	24845913
9호선	17801844
5호선	17150626
분당선	9838150
공항철도 1호선	8819860
경원선	7322160
8호선	6961678
6호선	5850424
경의선	2789012
신림선	345446

3 데이터 분석 및 시각화 – 역

상위 25개 역의 노선 중 상위 5개 노선 구하기

```
top25.groupby("노선").sum().sort_values(by="총 승객", ascending=False)[["총 승객"]].head(5)
```

노선	총 승객
2호선	316925409
경부선	63131791
1호선	47405460
7호선	37742970
3호선	29010952

```
top5_line=top25.groupby("노선").sum().sort_values(by="총 승객", ascending=False)[["총 승객"]].head(5).index
top5_line
```

```
Index(['2호선', '경부선', '1호선', '7호선', '3호선'], dtype='object', name='노선')
```

상위 25개 역 중 상위 5개 노선 리스트

상위 25개 역과 상위 5개 노선 결합

```
top=df9[top25_station_cond & top5_line_cond]
top
```

	날짜	노선	역	승차승객	하차승객	총 승객	승차-하차	월	일	요일
8	2022-01-01	1호선	서울역	18398	16926	35324	1472	1	1	토
10	2022-01-01	1호선	시청	5604	5006	10610	598	1	1	토
12	2022-01-01	1호선	종로3가	11017	9630	20647	1387	1	1	토
13	2022-01-01	2호선	구로디지털단지	18363	18352	36715	11	1	1	토
15	2022-01-01	2호선	신림	29564	29162	58726	402	1	1	토
...	...	...	...	...	...	...	...	...	...	...
145516	2022-08-31	경부선	용산	36725	37759	74484	-1034	8	31	수
145519	2022-08-31	경부선	영등포	39567	40820	80387	-1253	8	31	수
145520	2022-08-31	경부선	신도림	4427	4459	8886	-32	8	31	수
145523	2022-08-31	경부선	가산디지털단지	19472	22678	42150	-3206	8	31	수
145534	2022-08-31	경부선	수원	37197	39373	76570	-2176	8	31	수

상위 역과 노선의 결합

3 데이터 분석 및 시각화 – 역

상위 25개 역의 월별 총 승객 수 ¶

```
month_top=top.pivot_table(index="월", columns="역", values="총 승객")
month_top
```

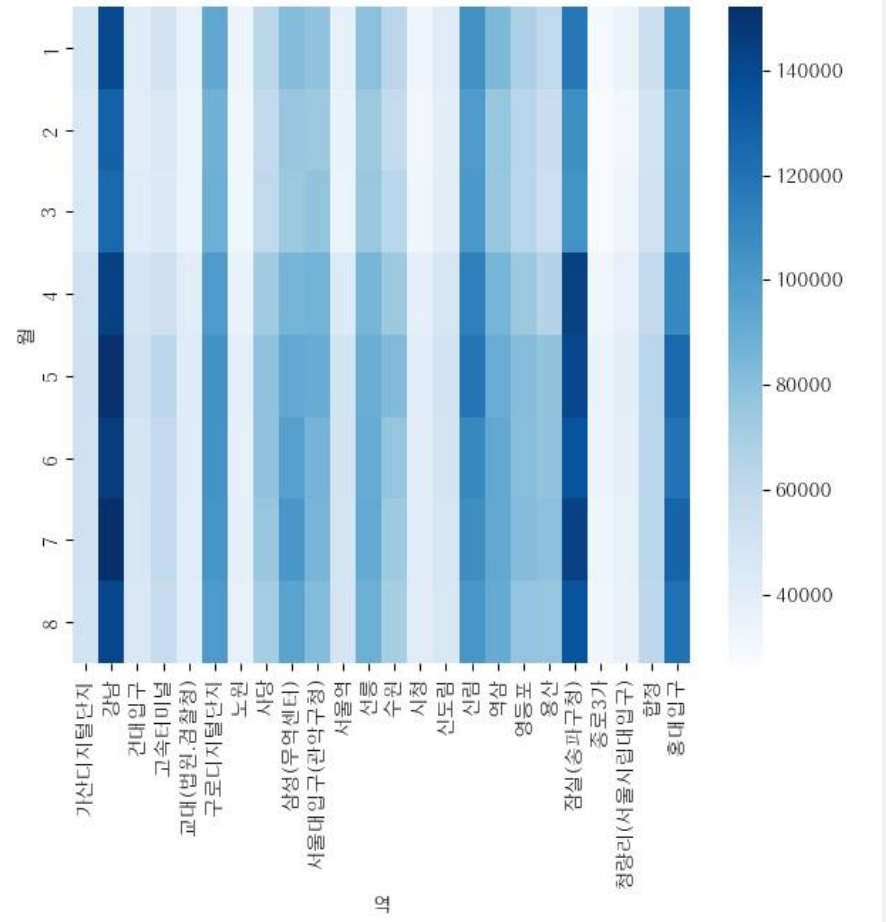
히트맵으로 확인

피벗 테이블로 확인

역 / 월	가산디지털단지	강남	건대입구	고속터미널	교대(법원.검찰청)	구로디지털단지	노원	사당	삼성(무역센터)	서울대입구(관악구청)	...	신도림	신림	역삼	영등포	
1	48587.887097	140046.612903	42505.016129	50003.500000	36280.290323	93925.548387	33571.741935	63003.032258	82130.193548	77890.774194	...	42397.064516	104728.838710	83323.193548	68113.580645	
2	45137.178571	128262.821429	39300.339286	46510.428571	33944.125000	87935.321429	31988.214286	59107.535714	75705.107143	74208.500000	...	40164.214286	100276.642857	76228.964286	63825.571429	
3	46057.193548	125027.064516	41509.935484	44173.854839	34828.935484	88651.774194	32111.612903	60258.838710	74310.580645	77404.419355	...	40642.741935	101544.548387	75427.096774	63353.064516	
4	51919.583333	143798.800000	48362.050000	54316.583333	40238.100000	99938.433333	35636.066667	72240.700000	85671.600000	86670.966667	...	47584.216667	113709.066667	85905.433333	74133.333333	
5	53113.741935	151591.322581	51905.419355	61968.387097	42570.548387	104746.419355	38403.483871	78781.322581	92969.419355	90625.870968	...	50523.854839	118474.645161	90154.451613	81111.064516	
6	52802.100000	146197.533333	48429.233333	59985.483333	42181.850000	104675.600000	37535.033333	78253.333333	97120.600000	86271.100000	...	49713.733333	109221.700000	92652.266667	80753.166667	
7	52252.951613	152091.838710	47790.838710	60341.677419	41294.903226	103647.580645	38293.774194	75767.193548	102367.838710	85625.129032	...	48029.677419	107071.290323	92616.935484	81603.806452	
8	50635.500000	140858.967742	45542.370968	57392.709677	39749.935484	100326.967742	36570.838710	71206.774194	95141.677419	82130.032258	...	46219.709677	102630.612903	91164.774194	77179.451613	

```
plt.subplots(figsize=(7, 6))
sns.heatmap(month_top, cmap='Blues')
```

```
<AxesSubplot:xlabel='역', ylabel='월'>
```

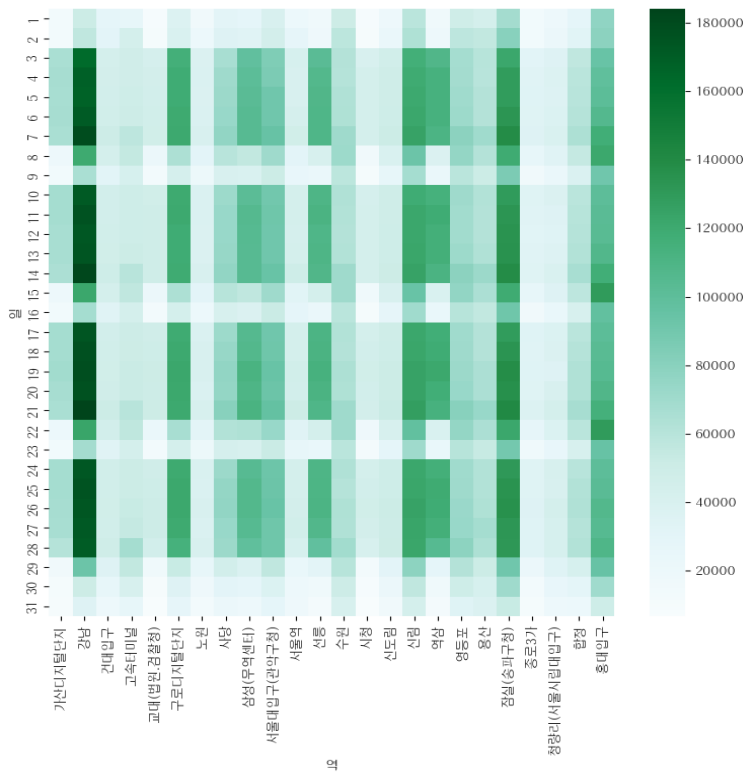


결과확인

- 강남과 잠실(송파구청)역의 총 승객이 눈에 띄게 많다는 것을 알 수 있다
- 교대(법원,검찰청)역과 노원역, 시청역, 종로3가, 청량리(서울시립대입구)역은 상대적으로 승객이 적은 것을 알 수 있다
- 25곳의 역 모두 2~3월 승객의 수가 적고, 5월과 7월이 승객의 수가 많다는 것을 알 수 있다

3 데이터 분석 및 시각화 – 역

상위데이터 월 별 히트맵 - 1월과 5월로 확인

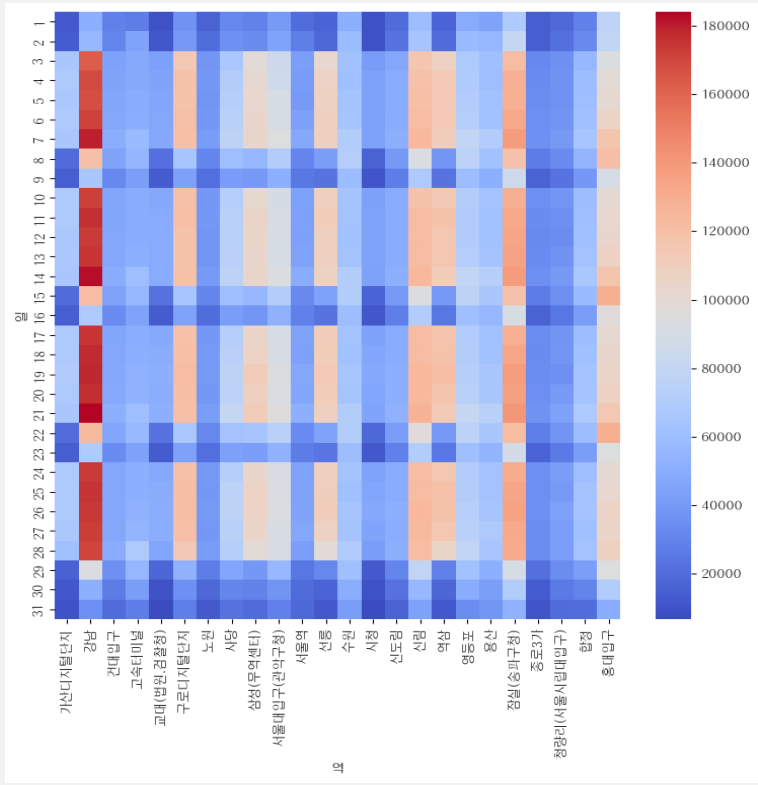


1월 히트맵

1월
- 주중에는 승객 많고, 주말은 적음
- 29~31일은 설 연휴 영향으로 사람 적은 것으로 판단

5월
- 화~토 승객 많고, 일~월 적음
- 연휴 기간의 영향으로 판단

공통적으로
홍대입구역은 상대적으로 고르게 나타남



5월 히트맵

3 데이터 분석 및 시각화 – 역

환승역으로 표현한 워드 클라우드

```
station_code['환승역 수'] = station_code.역.map(s)
station_code
```

	역코드	역	환승역 수
0	0	4.19민주묘지	1
1	1	가능	1
2	2	가락시장	2
3	3	가산디지털단지	2
4	4	가양	1
...	...	...	...
524	524	회기	1
525	525	회룡	1
526	526	회현(남대문시장)	1
527	527	효창공원앞	2
528	528	흑석(중앙대입구)	1

```
wc=station_code.set_index("역").to_dict()["환승역 수"]
```

```
wc2=df9.set_index("역").to_dict()["총 승객"]
```

총 승객으로 표현한 워드 클라우드

4 데이터 예측 – 데이터 세팅

자료 수정

- 요일, 노선을 int형으로 변환
- 날짜, 역 삭제

예측을 위해
Object 형 데이터의 수정 필요

Object 형 데이터
- 요일, 노선을 int 형으로 변환
- 날짜, 역 삭제

요일, 노선을 int형으로 바꾸기 위한 리스트 생성

```
week_mapping={"월":0, "화":1, "수":2, "목":3, "금":4, "토":5, "일":6}
```

```
line_mapping={"1호선":0, "2호선":1, "3호선":2, "4호선":3, "5호선":4, "6호선":5, "7호선":6, "8호선":7, "9호선":8, "9호선2~3단계":9, "경강선":10, "경부선":11, "경원선":12,
              "경의선":13, "경인선":14, "경춘선":15, "공항철도 1호선":16, "과천선":17, "분당선":18, "수인선":19, "신림선":20, "안산선":21, "우이신설선":22,
              "일산선":23, "장항선":24, "중앙선":25}
```

적용

```
df9["요일"]=df9["요일"].map(week_mapping)
```

```
df9["노선"]=df9["노선"].map(line_mapping)
```

```
df9=df9.drop(["날짜", "역"], axis=1)
```

```
df9.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145794 entries, 0 to 145793
Data columns (total 8 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------  -----
 0   노선      145794 non-null  int64
 1   승차승객    145794 non-null  int64
 2   하차승객    145794 non-null  int64
 3   총 승객    145794 non-null  int64
 4   승차-하차   145794 non-null  int64
 5   월       145794 non-null  int64
 6   일       145794 non-null  int64
 7   요일      145794 non-null  int64
dtypes: int64(8)
```

수정 완료

	노선	승차승객	하차승객	총 승객	승차-하차	월	일	요일
0	2	7370	7076	14446	294	1	1	5
1	2	461	473	934	-12	1	1	5
2	2	3224	2903	6127	321	1	1	5
3	2	3321	2803	6124	518	1	1	5
4	12	1	0	1	1	1	1	5
...	...	...	...	...	...	...	...	...
145789	25	901	848	1749	53	8	31	2
145790	25	639	632	1271	7	8	31	2
145791	25	321	317	638	4	8	31	2
145792	25	2850	2848	5698	2	8	31	2
145793	25	1754	1726	3480	28	8	31	2

기존 데이터를
훈련 데이터로 선정

4 데이터 예측 – 데이터 세팅

테스트 데이터 선정 및 세팅

```
df=pd.read_csv("Metro/2208-09.csv", encoding="euc-kr")
```

서울 열린데이터 광장의
지하철 호선별 데이터에서
최근 1달간 데이터 선정

데이터 변경 후 확인

	사용일자	호선명	역명	승차총승객수	하차총승객수	등록일자
0	20220925	우이신설선	4.19민주묘지	2283	1940	20220928
1	20220925	경원선	가능	4697	4287	20220928
2	20220925	8호선	가락시장	3454	4242	20220928
3	20220925	3호선	가락시장	4272	4510	20220928
4	20220925	경부선	가산디지털단지	5741	7522	20220928
...	...	...	...	...	...	...
23631	20220818	경원선	회룡	12956	12871	20220821
23632	20220818	4호선	회현(남대문시장)	25516	26182	20220821
23633	20220818	경의선	효창공원앞	3026	3405	20220821
23634	20220818	6호선	효창공원앞	7301	6515	20220821
23635	20220818	9호선	흑석(중앙대입구)	8399	8757	20220821

8월 데이터 삭제 필요

9월 데이터만 적용

```
df=df[df['월'] ==9]
```

```
df
```

	노선	승차승객	하차승객	총 승객	승차-하차	월	일	요일
0	22	2283	1940	4223	343	9	25	6
1	12	4697	4287	8984	410	9	25	6
2	7	3454	4242	7696	-788	9	25	6
3	2	4272	4510	8782	-238	9	25	6
4	11	5741	7522	13263	-1781	9	25	6
...	...	...	...	...	...	...	...	...
15141	12	14032	13972	28004	60	9	1	3
15142	3	27034	28278	55312	-1244	9	1	3
15143	13	3641	3776	7417	-135	9	1	3
15144	5	8109	7775	15884	334	9	1	3
15145	8	11397	11501	22898	-104	9	1	3

4 데이터 예측 – 데이터 정규화

훈련 데이터의 기술통계

	노선	승차승객	하차승객	총 승객	승차-하차	월	일	요일
count	145794.00000	145794.00000	145794.00000	145794.00000	145794.00000	145794.00000	145794.00000	145794.00000
mean	10.00296	9826.45728	9791.57458	19618.03186	34.88269	4.54116	15.70644	2.99512
std	7.05350	9877.38427	10016.66844	19864.62382	1090.62087	2.29835	8.79173	2.00688
min	0.00000	1.00000	0.00000	1.00000	-20299.00000	1.00000	1.00000	0.00000
25%	4.00000	3333.00000	3223.00000	6582.00000	-206.00000	3.00000	8.00000	1.00000
50%	9.00000	7000.50000	6821.50000	13852.00000	63.00000	5.00000	16.00000	3.00000
75%	15.00000	12953.00000	12796.00000	25805.75000	369.00000	7.00000	23.00000	5.00000
max	25.00000	122543.00000	118237.00000	240780.00000	28118.00000	8.00000	31.00000	6.00000

총 승객 값 분리

```
month_labels=month_df.pop("총 승객")
predict_labels=predict_df.pop("총 승객")
```

행과 열 변환

```
month_stats=month_stats.transpose()
month_stats
```

	count	mean	std	min	25%	50%	75%	max
노선	145794.00000	10.00296	7.05350	0.00000	4.00000	9.00000	15.00000	25.00000
승차승객	145794.00000	9826.45728	9877.38427	1.00000	3333.00000	7000.50000	12953.00000	122543.00000
하차승객	145794.00000	9791.57458	10016.66844	0.00000	3223.00000	6821.50000	12796.00000	118237.00000
승차-하차	145794.00000	34.88269	1090.62087	-20299.00000	-206.00000	63.00000	369.00000	28118.00000
월	145794.00000	4.54116	2.29835	1.00000	3.00000	5.00000	7.00000	8.00000
일	145794.00000	15.70644	8.79173	1.00000	8.00000	16.00000	23.00000	31.00000
요일	145794.00000	2.99512	2.00688	0.00000	1.00000	3.00000	5.00000	6.00000

데이터 정규화

```
def norm(x):
    return(x-month_stats["mean"])/month_stats["std"]
normed_month_df=norm(month_df)
normed_predict_df=norm(predict_df)
```

```
normed_month_df.tail()
```

	노선	승차승객	하차승객	승차-하차	월	일	요일
145789	2.12618	-0.90363	-0.89287	0.01661	1.50492	1.73954	-0.49586
145790	2.12618	-0.93015	-0.91443	-0.02557	1.50492	1.73954	-0.49586
145791	2.12618	-0.96235	-0.94588	-0.02832	1.50492	1.73954	-0.49586
145792	2.12618	-0.70631	-0.69320	-0.03015	1.50492	1.73954	-0.49586
145793	2.12618	-0.81727	-0.80522	-0.00631	1.50492	1.73954	-0.49586

```
normed_predict_df.tail()
```

	노선	승차승객	하차승객	승차-하차	월	일	요일
15141	0.28313	0.42577	0.41735	0.02303	1.94001	-1.67276	0.00243
15142	-0.99283	1.74212	1.84557	-1.17262	1.94001	-1.67276	0.00243
15143	0.42490	-0.62622	-0.60056	-0.15577	1.94001	-1.67276	0.00243
15144	-0.70929	-0.17388	-0.20132	0.27426	1.94001	-1.67276	0.00243
15145	-0.28397	0.15900	0.17066	-0.12734	1.94001	-1.67276	0.00243

4 데이터 예측 – 모델 훈련

```
# 모델 만들기 – mae(평균 절대 오차), mse(평균 제곱 오차)
def build_model():
    model=keras.Sequential([
        layers.Dense(64, input_shape=[len(month_df.keys())], activation="relu"),
        layers.Dense(64, activation="relu"),
        layers.Dense(1)
    ])
    # 오답 처리 알고리즘 – 바로 밑
    optimizer=tf.keras.optimizers.RMSprop(0.001)
    model.compile(loss="mse", optimizer=optimizer, metrics=["mae", "mse"])
    return model
```

```
# 모델 확인
model=build_model()
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                512

dense_1 (Dense)              (None, 64)                4160

dense_2 (Dense)              (None, 1)                 65

=================================================================
Total params: 4,737
Trainable params: 4,737
Non-trainable params: 0
_________________________________________________________________
```

```
# 모델 훈련 – 100번
class PrintDot(keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs):
        if epoch%100==0:
            print("")
        print('.', end="")
EPOCHS=100
model=build_model()
history=model.fit(
    normed_month_df, month_labels, epochs=EPOCHS, validation_split=0.2, verbose=0, callbacks=[PrintDot()]
)
```

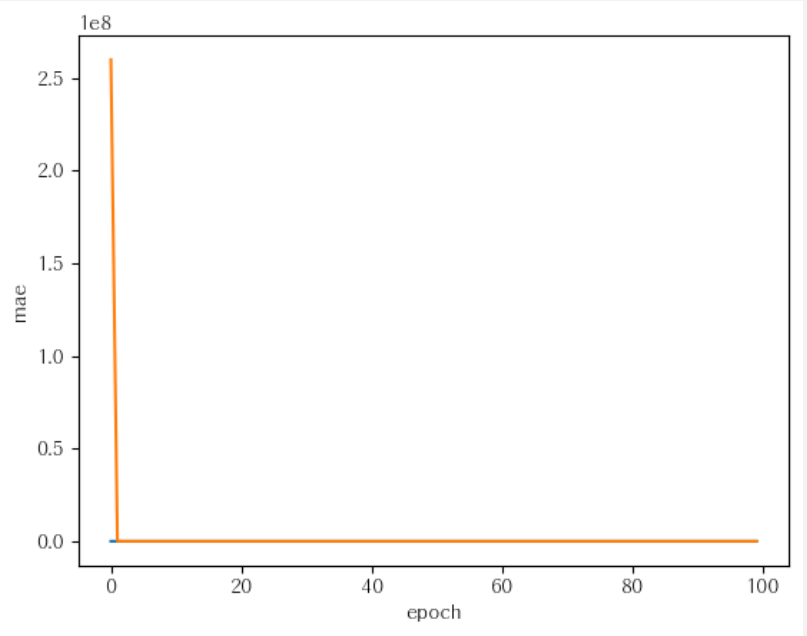
만들어진 모델을 100번 훈련

```
.....................................................................................................
```

```
hist=pd.DataFrame(history.history)
hist.tail()
```

	loss	mae	mse	val_loss	val_mae	val_mse
95	599.98914	10.46475	599.98914	1406.55042	24.41073	1406.55042
96	596.43951	10.85181	596.43951	40.41108	5.12634	40.41108
97	609.68500	10.52835	609.68500	3316.63867	34.79006	3316.63867
98	626.11035	10.36290	626.11035	642.83691	19.23706	642.83691
99	617.15411	10.54993	617.15411	15.72472	3.17295	15.72472

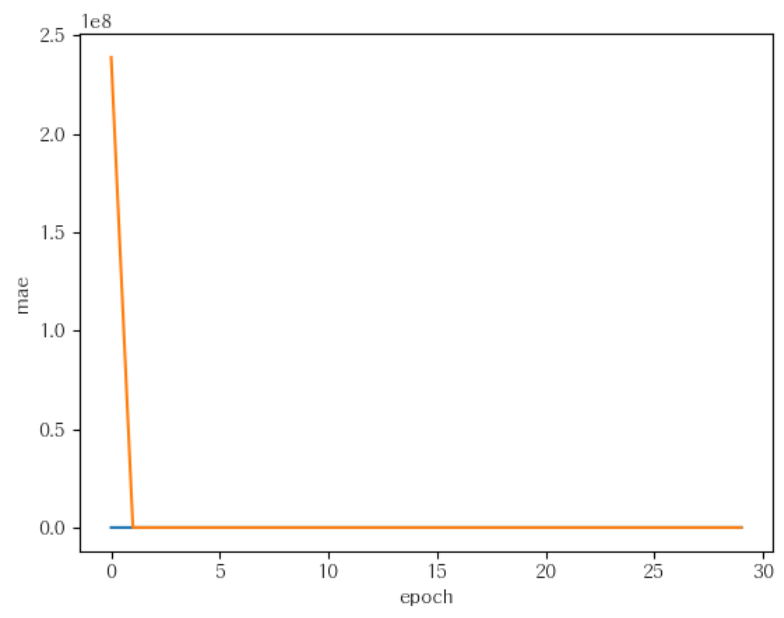
```
# 학습 100번 가성비
# 학습을 진행하면서 오차율을 시각화
def sns_hist(history):
    hist=pd.DataFrame(history.history)
    hist['epoch']=history.epoch
    sns.lineplot(data=hist, x="epoch", y="mae")
    sns.lineplot(data=hist, x="epoch", y="mse")
sns_hist(history)
```



4 데이터 예측 – 성능 평가

훈련 모델의 효율성 확인

```
# model.fit를 수정해서 검증 점수가 향상되지 않으면 자동으로 훈련을 멈추도록 하는 것으로 수정(val_loss기준)
model=build_model()
early_stop=keras.callbacks.EarlyStopping(monitor="val_loss", patience=10)
history=model.fit(
    normed_month_df, month_labels, epochs=EPOCHS, validation_split=0.2, verbose=0, callbacks=[PrintDot(), early_stop]
)
#verbose=0 – log 안 찍는 것
```



성능평가

```
# 성능 평가
loss, mae, mse=model.evaluate(normed_predict_df, predict_labels, verbose=2)
print("테스트 세트의 평균 절대 오차: ", mae)
print("테스트 세트의 평균 절대 오차: {:.5f}".format(mae))
```

```
474/474 - 0s - loss: 390.8442 - mae: 14.4918 - mse: 390.8442 - 213ms/epoch - 448us/step
테스트 세트의 평균 절대 오차:  14.491832733154297
테스트 세트의 평균 절대 오차: 14.49183
```

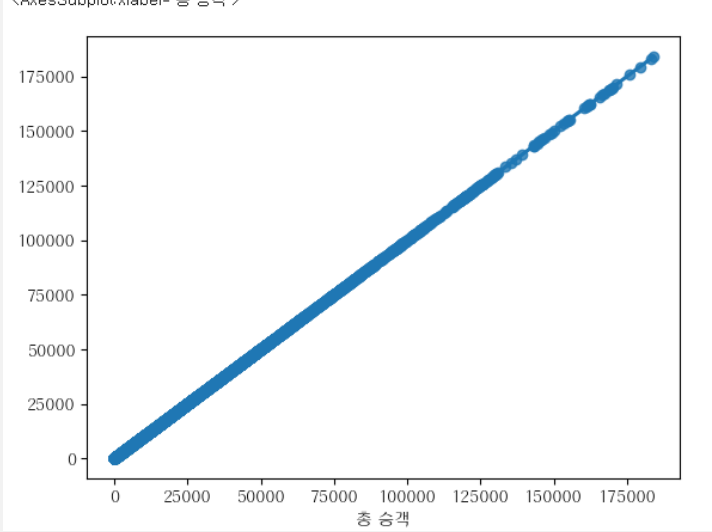
```
# 시각화해서 평가
# 예측만 하게 할 수 있음
predict_predictions=model.predict(normed_predict_df).flatten()
predict_predictions
```

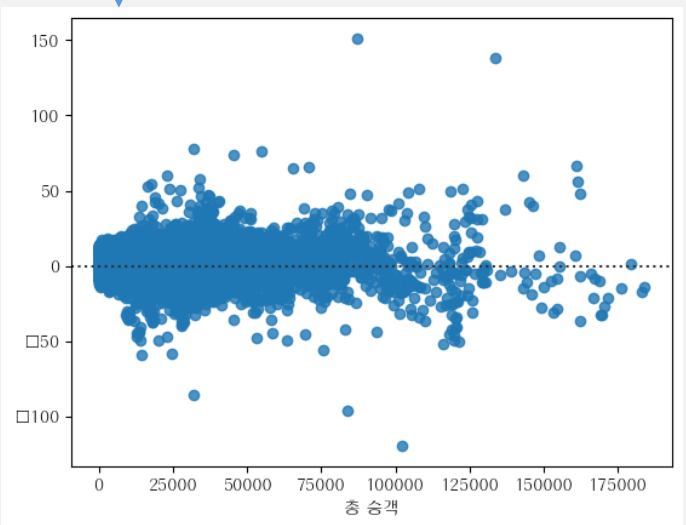
```
474/474 [==============================] - 0s 477us/step
array([ 4219.0225,  8983.896 ,  7702.543 , ...,  7434.8945, 15904.997 ,
       22923.945 ], dtype=float32)
```

시각화

```
sns.regplot(x=predict_labels, y=predict_predictions)
```

```
<AxesSubplot:xlabel='층 승객'>
```



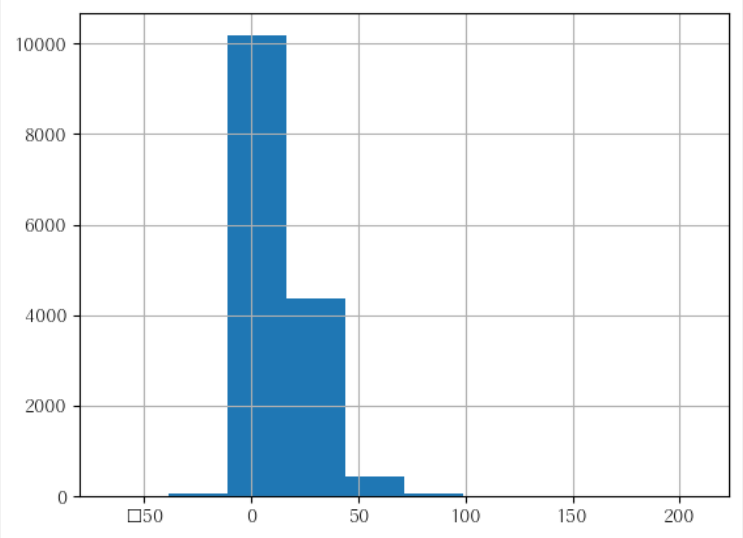


4 데이터 예측 – 평가 및 예측치

```
# 오차 분포
error=predict_predictions-predict_labels
error.tail()
```

```
15141   26.66016
15142   47.97266
15143   17.89453
15144   20.99707
15145   25.94531
Name: 총 승객, dtype: float64
```

시각화



k-NN 알고리즘으로 평가

```
# k-NN 분류와 회귀에 사용되는 알고리즘
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(X_train, Y_train)
Y_pred=knn.predict(X_test)
acc_knn=round(knn.score(X_train, Y_train)*100, 2)
acc_knn
```

```
13.37
```

※ 다른 알고리즘은
메모리 에러 발생

승차/하차 승객 예측

```
predict_month=pd.DataFrame({
    "승차승객": predict_df["승차승객"],
    "하차승객":Y_pred
    })
predict_month.head()
```

	승차승객	하차승객
0	2283	2257
1	4697	4681
2	3454	3456
3	4272	4253
4	5741	5652

5 추가 / 보완사항

1 메모리 에러 해결

데이터가 많아서 알고리즘으로 평가할 때,
k-NN을 제외한 알고리즘은 메모리 에러가
발생하므로, 해결방안을 찾아서
다른 알고리즘으로 교차 검증 가능케 해야 함

2 예측치 정밀 보완

데이터의 승차/하차 승객의 수를 예측하는 것은
성공했으나, 승차/하차 승객이 어떤 노선과 역을
이용했고, 언제 이용했는지는 파악하지 못함

3 데이터 세팅 보완

노선과 요일은 int 형으로 변환하여,
훈련 모델에 적용시켰으나, 역은 개수가
많아서 리스트로 만들지 못하여 예측의
정확도가 떨어짐

4 컬럼 간 상관관계

컬럼 간 상관관계를 파악하고 시각화는 했지만,
상관관계가 낮은 컬럼을 변환하여,
추가로 만들어서 적용하진 못 했음.

감 사 합 니 다