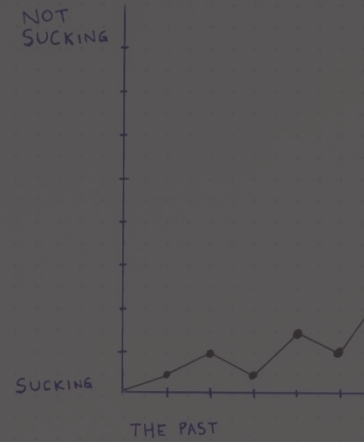


# 코로나 이후 지하철 수요 변화 및 역별 혼잡도 분석

1조 - 김정명, 박성하, 정길종, 허재혁



# 목 차

1

## 프로젝트 개요

- 프로젝트 기획 배경 및 목표
- 구성원 및 역할

2

## 프로세싱

- 데이터 수집
- 데이터 처리
- 데이터 저장

3

## 데이터 분석

- 데이터 분석
- 데이터 분석 결과

4

## 기대 효과

- 분석을 통한 인사이트 도출
- 향후 개선 사항 및 기대 효과

5

## 개발 후기 및 느낀점





The left side of the slide features a decorative background with overlapping geometric shapes in shades of orange, brown, and teal. In the background, there is a photograph of stationery items including two pens and a small white card.

---

Part 1.

# 프로젝트 개요

- 프로젝트 기획 배경 및 목표
- 구성원 및 역할

# 1. 프로젝트 개요

## 1) 프로젝트 기획 배경 및 목표

- **코로나 19 발생에 따라 여객 부분 수송실적 급감** (한국교통연구원, 2020)
    - 철도의 경우, 국내 첫 확진자 발생 전인 1월 3주차 대비 2월 1주차에 17%, 3월 1주차에 38% 감소
  - 이후 대규모 집단 감염과 지역확산이 지속적으로 발생 → 사회적 거리두기 단계를 격상하여 코로나 19 확산에 대응
  - 대중교통 측면에서 코로나 19 발발의 경제적 및 사회적 영향은 **교통 서비스 성과 감소**를 넘어 **재정적인 생존 가능성, 지속 가능한 이동성 위험** 등으로 확장될 수 있음
  - 이에 인구 밀집도가 높고 유동인구가 많은 서울시를 기준으로 지하철 수요 변화 및 혼잡도를 분석, 시각화하여 제공
- 고객들이 지하철을 보다 쾌적하고 안전하게 이용할 수 있도록 역 별로 적절한 정책 실시 기대

# 1. 프로젝트 개요

## 1) 프로젝트 기획 배경 및 목표

코로나 19 확산과 대중교통 수요 감소

대중교통 측면의 서울 시민 코로나 19 대응 양상 파악 및 예측 필요

재난 관리 및 수익 보전 측면에서 지하철 이용 변화 양상에 대한 구체적인 파악 필요



- 코로나 발생을 기점으로 일별, 시간대별 서울시 지하철 이용객수의 변화 분석
- 지하철 승하차인원 데이터를 바탕으로 역별 혼잡도 분석

# 1. 프로젝트 개요

## 2) 구성원 및 역할

- **조장 : 허재혁**
  - 데이터 수집, 분석 및 발표
- **팀원 1 : 김정명**
  - 데이터베이스 연동 및 구축
- **팀원 2 : 박성하**
  - 데이터 수집, 분석 및 시각화
- **팀원 3 : 정길종**
  - 데이터 수집, 시각화



---

## Part 2.

# 프로세싱

- 데이터 수집
- 데이터 처리
- 데이터 저장

# ELT Process

모든 데이터를 적재하고 처리하자!

Data Source

Data Lakes

Data Processing

Data Mart

DashBoard



보존



SQL



가공



시각화





# 2. 프로세싱

## 1) 데이터 수집 및 저장

### 서울시 지하철호선별 역별 승하차 인원 정보

교통카드(선후불교통카드 및 1회용 교통카드)를 이용한 지하철호선별 역별(서울교통공사, 한국철도공사, 공항철도, 9호선) 승하차인원을 나타내는 정보입니다. (일단위)

※ Sheet 서비스는 마지막 한달치 데이터만 서비스 합니다. (\* 데이터 적재는 매일 3일전 데이터를 갱신합니다.)

```
{
  "CardSubwayStatsNew": {
    "list_total_count": 602,
    "RESULT": {
      "CODE": "INFO-000",
      "MESSAGE": "정상 처리되었습니다.",
      "row": [
        {
          "USE_DT": "20210813",
          "LINE_NUM": "1호선",
          "SUB_STA_NM": "서울역",
          "RIDE_PASGR_NUM": 35591,
          "ALIGHT_PASGR_NUM": 37301,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "1호선",
          "SUB_STA_NM": "동묘앞",
          "RIDE_PASGR_NUM": 6867,
          "ALIGHT_PASGR_NUM": 7032,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "1호선",
          "SUB_STA_NM": "종로3가",
          "RIDE_PASGR_NUM": 28005,
          "ALIGHT_PASGR_NUM": 26970,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "1호선",
          "SUB_STA_NM": "종로5가",
          "RIDE_PASGR_NUM": 22407,
          "ALIGHT_PASGR_NUM": 20877,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "1호선",
          "SUB_STA_NM": "충무로",
          "RIDE_PASGR_NUM": 19667,
          "ALIGHT_PASGR_NUM": 19341,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "1호선",
          "SUB_STA_NM": "신설동",
          "RIDE_PASGR_NUM": 11972,
          "ALIGHT_PASGR_NUM": 11742,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "1호선",
          "SUB_STA_NM": "신설동",
          "RIDE_PASGR_NUM": 15023,
          "ALIGHT_PASGR_NUM": 15283,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "1호선",
          "SUB_STA_NM": "청량리(서울시립대입구)",
          "RIDE_PASGR_NUM": 17851,
          "ALIGHT_PASGR_NUM": 18259,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "시정",
          "RIDE_PASGR_NUM": 19306,
          "ALIGHT_PASGR_NUM": 18409,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "을지로입구",
          "RIDE_PASGR_NUM": 32609,
          "ALIGHT_PASGR_NUM": 33000,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "을지로3가",
          "RIDE_PASGR_NUM": 18336,
          "ALIGHT_PASGR_NUM": 18402,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "을지로4가",
          "RIDE_PASGR_NUM": 11553,
          "ALIGHT_PASGR_NUM": 11422,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "동대문역사문화공원(DDP)",
          "RIDE_PASGR_NUM": 9013,
          "ALIGHT_PASGR_NUM": 9779,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "신당",
          "RIDE_PASGR_NUM": 11729,
          "ALIGHT_PASGR_NUM": 12222,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "상왕십리",
          "RIDE_PASGR_NUM": 12454,
          "ALIGHT_PASGR_NUM": 11947,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "왕십리(성동구청)",
          "RIDE_PASGR_NUM": 13622,
          "ALIGHT_PASGR_NUM": 11093,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "한양대",
          "RIDE_PASGR_NUM": 5774,
          "ALIGHT_PASGR_NUM": 5615,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "독성",
          "RIDE_PASGR_NUM": 18813,
          "ALIGHT_PASGR_NUM": 20315,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "성수",
          "RIDE_PASGR_NUM": 31781,
          "ALIGHT_PASGR_NUM": 34900,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "건대입구",
          "RIDE_PASGR_NUM": 29089,
          "ALIGHT_PASGR_NUM": 31437,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "구의(광진구청)",
          "RIDE_PASGR_NUM": 20720,
          "ALIGHT_PASGR_NUM": 20135,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "강변(동서울터미널)",
          "RIDE_PASGR_NUM": 27067,
          "ALIGHT_PASGR_NUM": 28909,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "잠실(송파구청)",
          "RIDE_PASGR_NUM": 12663,
          "ALIGHT_PASGR_NUM": 12201,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "잠실(송파구청)",
          "RIDE_PASGR_NUM": 56330,
          "ALIGHT_PASGR_NUM": 55755,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "잠실(송파구청)",
          "RIDE_PASGR_NUM": 18194,
          "ALIGHT_PASGR_NUM": 18401,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "종합운동장",
          "RIDE_PASGR_NUM": 5692,
          "ALIGHT_PASGR_NUM": 5841,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "삼성(무역센터)",
          "RIDE_PASGR_NUM": 42693,
          "ALIGHT_PASGR_NUM": 42912,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "선릉",
          "RIDE_PASGR_NUM": 48259,
          "ALIGHT_PASGR_NUM": 43071,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "역삼",
          "RIDE_PASGR_NUM": 42982,
          "ALIGHT_PASGR_NUM": 47070,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "강남",
          "RIDE_PASGR_NUM": 73098,
          "ALIGHT_PASGR_NUM": 71055,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "교대(병원 경할)",
          "RIDE_PASGR_NUM": 29777,
          "ALIGHT_PASGR_NUM": 33837,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "서초",
          "RIDE_PASGR_NUM": 18964,
          "ALIGHT_PASGR_NUM": 18192,
          "WORK_DT": "20210816",
          "USE_DT": "20210813",
          "LINE_NUM": "2호선",
          "SUB_STA_NM": "방배"
        }
      ]
    }
  }
}
```

1. Request

2. Response

3. Database : datalake  
Collection : i  
처리하지 않고 바로 저장

## Open API – MongoDB 저장

```
from urllib.request import Request, urlopen
from urllib.parse import urlencode, quote_plus
from xml.etree import ElementTree
```

```
from datetime import datetime, timedelta
from pymongo import MongoClient
import pandas as pd
import json
```

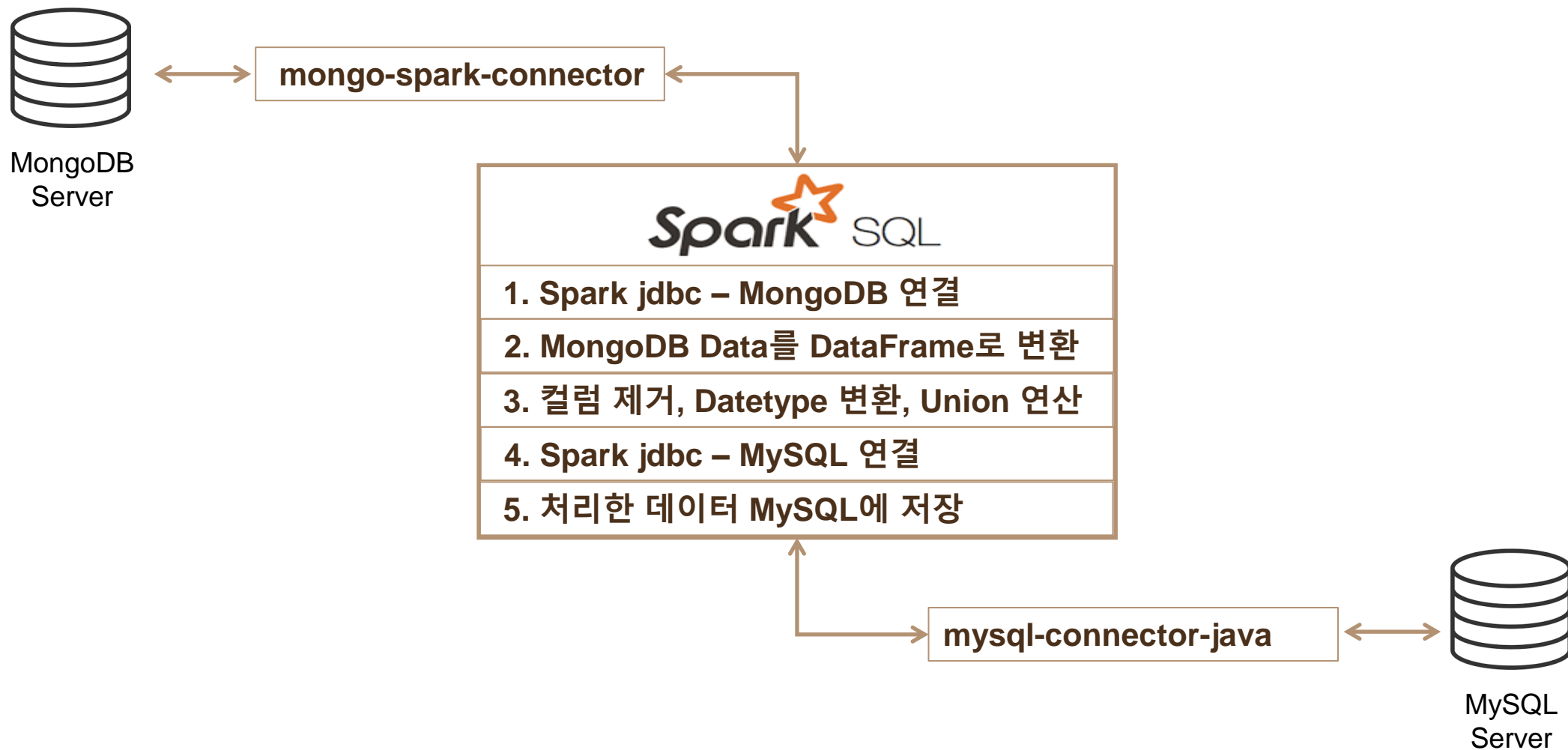
```
url = "http://openapi.seoul.go.kr:8088/{0}/json/CardSubwayStatsNew/1/1000/".format(skey)
skey = "5752524157776a643837426953524b"
today = datetime.today() - timedelta(4)
today_4 = today.strftime("%Y%m%d")
response_body = urlopen(url)
body = json.loads(response_body.read())
responseBody = urlopen(url+today_4).read().decode('utf-8')
jsonArray = json.loads(responseBody)
storeInfosArray= jsonArray["CardSubwayStatsNew"]["row"]
```

```
# pymongo connect
client = MongoClient('localhost',27017) # mongodb 27017 port
db = client.datalake
if storeInfosArray is not None:
    for j in range(len(storeInfosArray)):
        db.i.insert_one(storeInfosArray[j])
```

Open API / 주기 1일 / DataFormat : json

## 2. 프로세싱

### 2) 데이터 처리 및 저장



# 2. 프로세싱

## 3) Ubuntu Scheduler crontab

### 전체 수동버전

```
## 전체수동버전

## Data_collection.py
from urllib.request import Request, urlopen
from urllib.parse import urlencode, quote_plus
from xml.etree import ElementTree

from pyspark.conf import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType
from pyspark.sql.functions import array_contains, udf

from datetime import datetime, timedelta
from pymongo import MongoClient
import pandas as pd
import json

## processing_load.py
spark = SparkSession \
    .builder \
    .appName("multi") \
    .config("spark.mongodb.input.uri", "mongodb://localhost:27017") \
    .config("spark.mongodb.input.database", "datalake") \
    .config("spark.mongodb.input.collection", "i") \
    .config("packages org.mongodb.spark:mongo-spark-connector_2.12:3.0.1") \
    .getOrCreate()
sc = spark.sparkContext

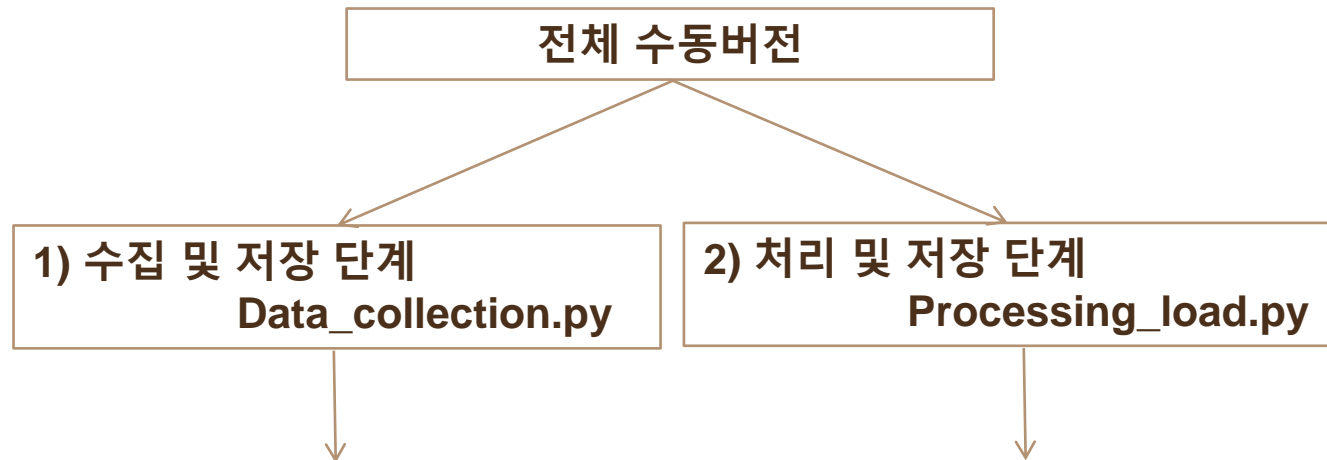
# pymongo connect
client = MongoClient('localhost', 27017) # mongodb 27017 port
db = client.datalake

skey = "5752524157776a643837426953524b"
today = datetime.today() - timedelta(4)
today_4 = today.strftime("%Y%m%d")

url = "http://openapi.seoul.go.kr:8088/{0}/json/CardSubwayStatsNew/1/1000/".format(skey)

response_body = urlopen(url)
body = json.loads(response_body.read())
responseBody = urlopen(url+today_4).read().decode('utf-8')
```

### Batch Scheduling



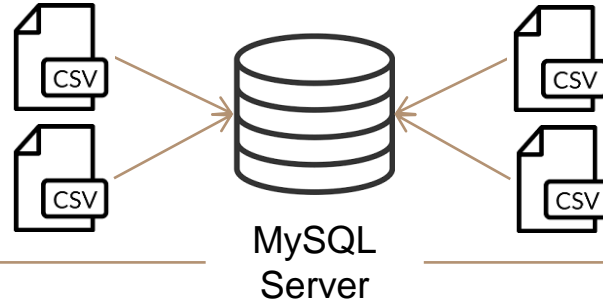
### crontab

```
ubuntu@ip-172-31-10-132:~$ crontab -l
#!/bin/bash
00 10 * * * ~/anaconda3/envs/py38_multi/bin/python Data_collection.py
01 10 * * * /opt/spark/bin/spark-submit processing_load.py
```

- 아침 10시에 open API 에서 데이터수집 후 몽고db에 저장
- 1분 뒤 데이터 처리(column 제거, date type 변환) 후 mysql에 저장

## 2. 프로세싱

### 4) MySQL 저장 확인 및 tableau 접속



A screenshot of the MySQL Workbench interface. The left sidebar shows a 'SCHEMAS' tree with a 'proj' database selected. The main window displays a SQL query: `SELECT * FROM proj.conf_20_21;`. Below the query, a 'Result Grid' shows a table with columns: month, line, st, usable\_area, under5, under6, under7, under8, under9, under10, under11, under12, under13, and under14. The data is organized by month (202107, 202108, 202109, 202110, 202111, 202112, 202201).

month	line	st	usable_area	under5	under6	under7	under8	under9	under10	under11	under12	under13	under14
202107	1호선	동대문	732	0.003	0.07	0.07	0.116	0.189	0.173	0.155	0.163	0.168	0.1
202107	1호선	동대문	1468	0	0.01	0.019	0.032	0.067	0.054	0.06	0.078	0.088	0.0
202107	1호선	서울역	728	0.003	0.069	0.208	0.524	0.99	0.647	0.396	0.385	0.434	0.4
202107	1호선	시정	1324	0	0.016	0.057	0.161	0.403	0.187	0.088	0.09	0.087	0.0
202107	1호선	신설동	1077	0.001	0.034	0.058	0.123	0.254	0.145	0.105	0.097	0.097	0.1
202107	1호선	제기동	985	0.001	0.025	0.061	0.136	0.237	0.197	0.215	0.223	0.217	0.2
202107	1호선	종각	935	0	0.022	0.091	0.327	0.729	0.438	0.222	0.222	0.215	0.2
202107	1호선	종로3가	1289	0.001	0.017	0.043	0.078	0.193	0.214	0.194	0.214	0.217	0.2
202107	1호선	종로5가	773	0	0.023	0.08	0.186	0.451	0.313	0.283	0.302	0.297	0.3
202107	1호선	정발리	620	0.005	0.084	0.188	0.267	0.406	0.323	0.332	0.35	0.355	0.3
202107	2호선	강남	831	0	0.066	0.257	0.721	1.419	1.37	0.729	0.657	0.707	0.8
202107	2호선	강변	827	0	0.04	0.18	0.382	0.561	0.4	0.302	0.277	0.288	0.2
202107	2호선	건대입구	901	0.001	0.058	0.143	0.27	0.504	0.368	0.243	0.241	0.279	0.3
202107	2호선	고대	1052	0	0.029	0.114	0.243	0.569	0.482	0.263	0.238	0.237	0.2
202107	2호선	구로디지털단지	820	0.002	0.185	0.295	0.846	1.656	0.942	0.434	0.362	0.382	0.4

(Local) MySQL Workbench 접속 결과

영결

분기

seoul\_st\_lining\_line\_thanks

시도별

subway\_2021

시도별

3:35 04:153

19/05

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

시도별

데이터베이스

(Local) Tableau 접속 결과

# 2. 프로세싱

## 5) 추가 데이터 처리작업

```
    .set("spark.driver.extraClassPath", "/opt/spark/jars/mysql-connector-java-8.0.26.jar")
sc = SparkContext(conf=conf)

sqlCtx = SQLContext(sc)
spark = sqlCtx.sparkSession

sql_url = "localhost"
user = "root"
password = "1234"
database = "proj"
```

```
sub2019 = spark.read.format("jdbc")#
    .option("driver", "com.mysql.jdbc.Driver")#
    .option("url", "jdbc:mysql://{ }:3306/{ }?serverTimezone=Asia/Seoul ".format(sql_url, database))#
    .option("user", user)#
    .option("password", password)#
    .option("dbtable", 'sub2019')#
    .load()
```

```
sub2020 = spark.read.format("jdbc")#
    .option("driver", "com.mysql.jdbc.Driver")#
    .option("url", "jdbc:mysql://{ }:3306/{ }?serverTimezone=Asia/Seoul ".format(sql_url, database))#
    .option("user", user)#
    .option("password", password)#
    .option("dbtable", 'sub2020')#
    .load()
```

```
sub2021 = spark.read.format("jdbc")#
    .option("driver", "com.mysql.jdbc.Driver")#
    .option("url", "jdbc:mysql://{ }:3306/{ }?serverTimezone=Asia/Seoul ".format(sql_url, database))#
    .option("user", user)#
    .option("password", password)#
    .option("dbtable", 'sub2021')#
    .load()
```

```
sub2019 = sub2019.drop('24over')
sub2020 = sub2020.drop('24over')
```

```
def unionAll(*dfs):
    return reduce(DataFrame.union, dfs)

sub_tot = unionAll(sub2019, sub2020, sub2021)
```

```
dateFormat = "yyyy-MM-dd"
sub_tot = sub_tot.withColumn("date", f.to_date(f.unix_timestamp(sub_tot.date, dateFormat).cast('timestamp')))
```

```
sub_tot.write #
    .format("jdbc") #
    .option("url", "jdbc:mysql://localhost/proj") #
    .option("dbtable", "proj.sub_total") #
    .option("user", "root") #
    .option("password", "1234") #
    .save()
```

ex) MySQL data에서 sub2019, sub2020, sub2021 데이터를 통합



1. Spark와 MySQL Connector를 통해 3개의 sparkContext 생성

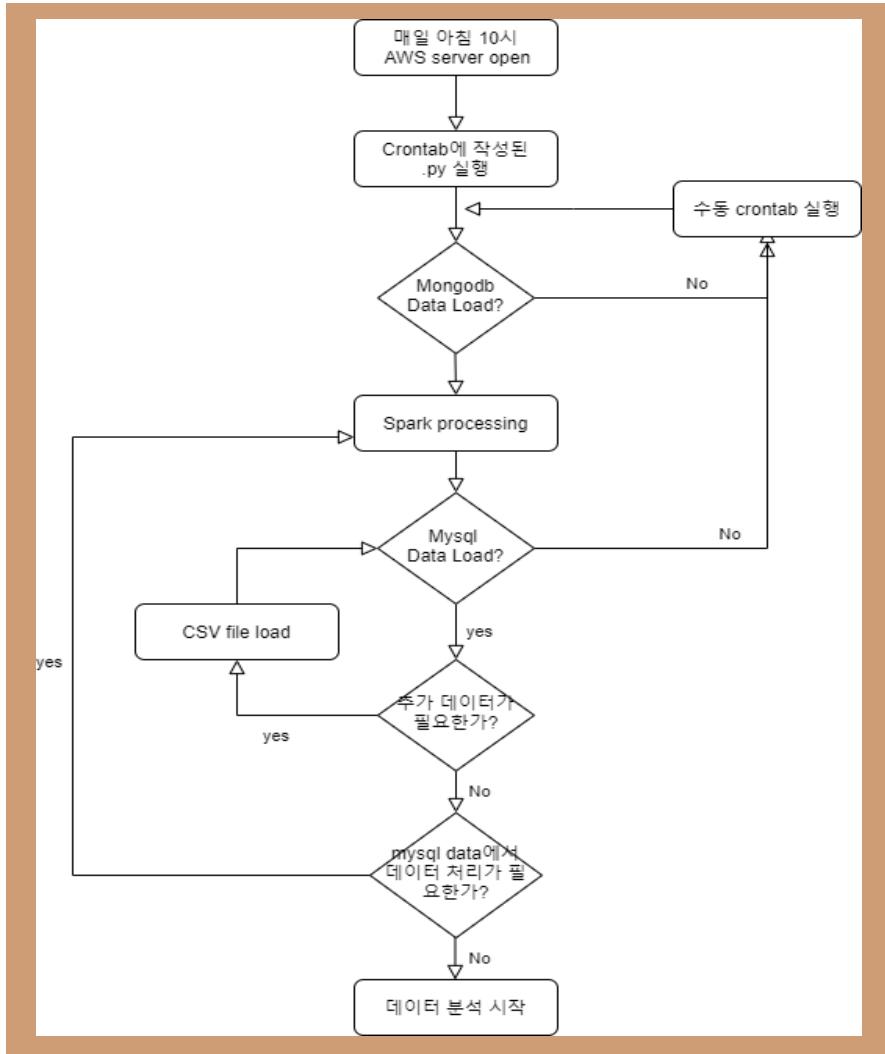
2. Column 제거 후 UDF를 이용한 Union 함수를 구현하여 join

3. 하나의 테이블로 join 후 MySQL에 저장



## 2. 프로세싱

### 6) FlowChart / 테이블 정의서



Entity 명		subway_st_info	
Entity 정의		역내 면적 및 혼잡 면적	
No	Column 명	한글명	자료형
1	line	호선	varchar(10)
2	st	역이름	varchar(10)
3	haehasil	대합실 면적	Double
4	seunggang	승강장 면적	Double
5	usable_are	가용가능 면적	Double
6	LOS B	B기준 면적	Double
7	LOS C	C기준 면적	Double
8	LOS D	D기준 면적	Double
9	LOS E	E기준 면적	Double

Entity 명		subway_dist	
Entity 정의		역간 거리	
No	Column 명	한글명	자료형
1	line	호선	varchar(10)
2	st	역이름	varchar(10)
3	bet_dist	역간 거리	Float
4	sum_dist	누적 거리	Float

Entity 명		subway_total	
Entity 정의		시간대별 지하철 승하차 인원	
No	Column 명	한글명	자료형
1	date	날짜	DATE
2	line	호선	TEXT
3	st_nm	역 번호	INT(11)
4	st	역 이름	TEXT
5	gubun	승하차 구분	TEXT
6	under6	6시 이전	INT(11)
7	under7	7시 이전	INT(11)
8	under8	8시 이전	INT(11)
9	under9	9시 이전	INT(11)
10	under10	10시 이전	INT(11)
11	under11	11시 이전	INT(11)
12	under12	12시 이전	INT(11)
13	under13	13시 이전	INT(11)
14	under14	14시 이전	INT(11)
15	under15	15시 이전	INT(11)
16	under16	16시 이전	INT(11)

(테이블 정의서.xlsx)



---

## Part 3.

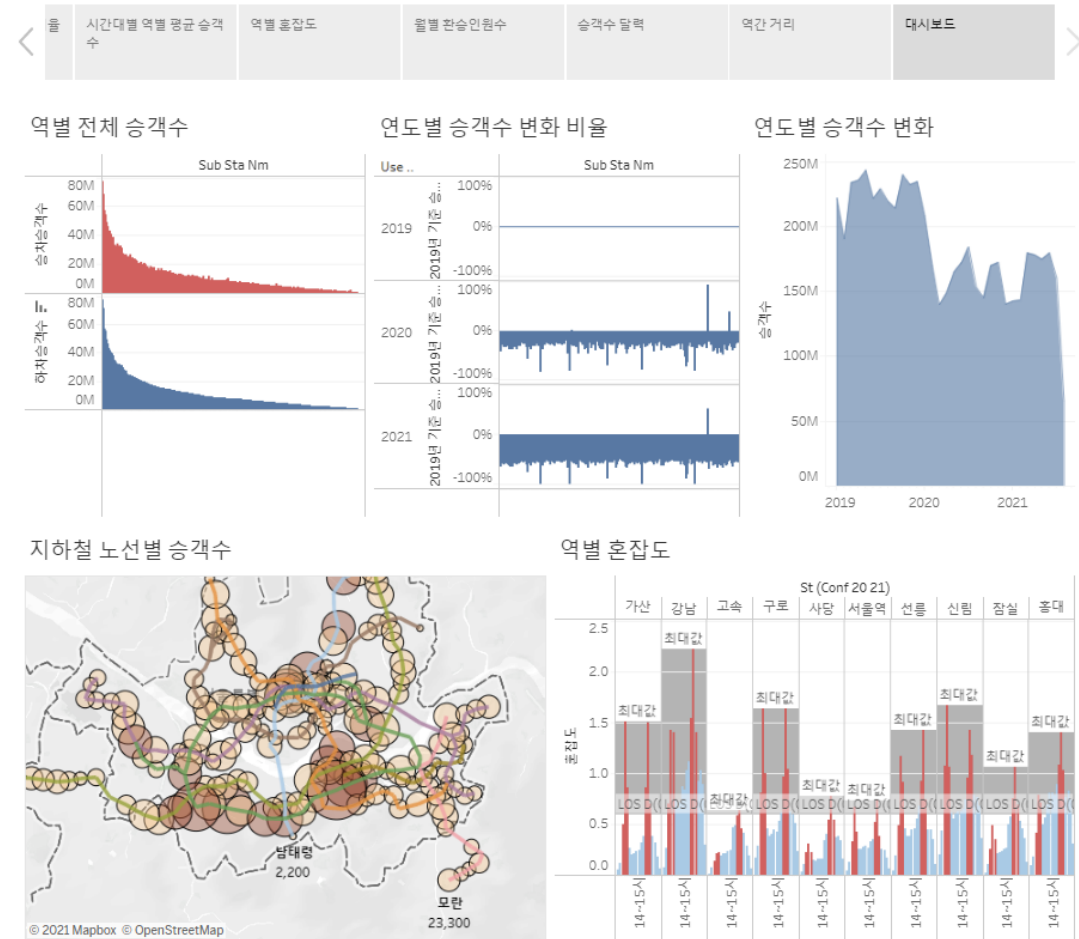
# 데이터 분석

- 데이터 분석
- 데이터 분석 결과

# 3. 데이터 분석

## 0) 대시보드

서울 지하철 비교 분석



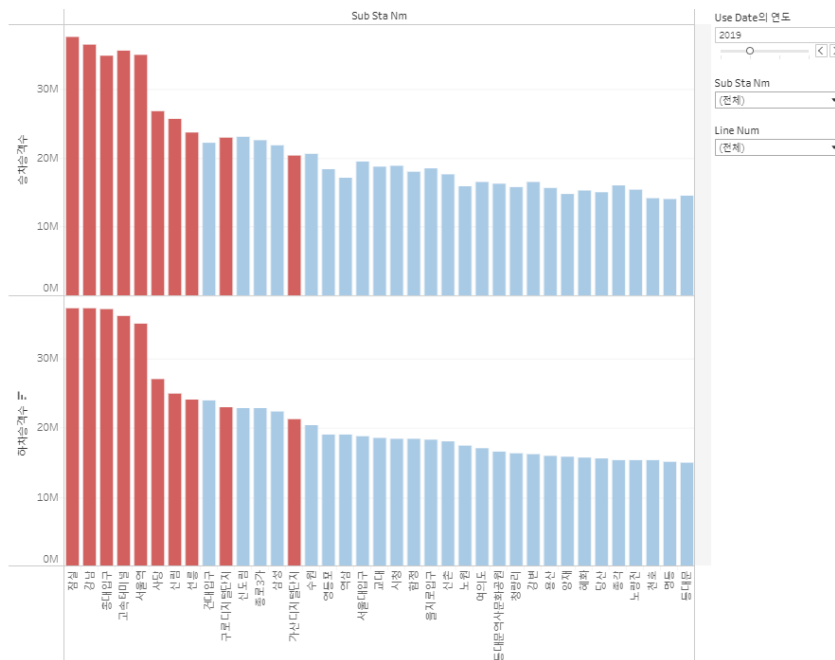
### ■ 시각화한 데이터

- 역별 전체 승객수
- 연도별 승객수 변화 비율
- 연도별 승객수 변화
- 지하철 노선별 승객수
- 역별 혼잡도

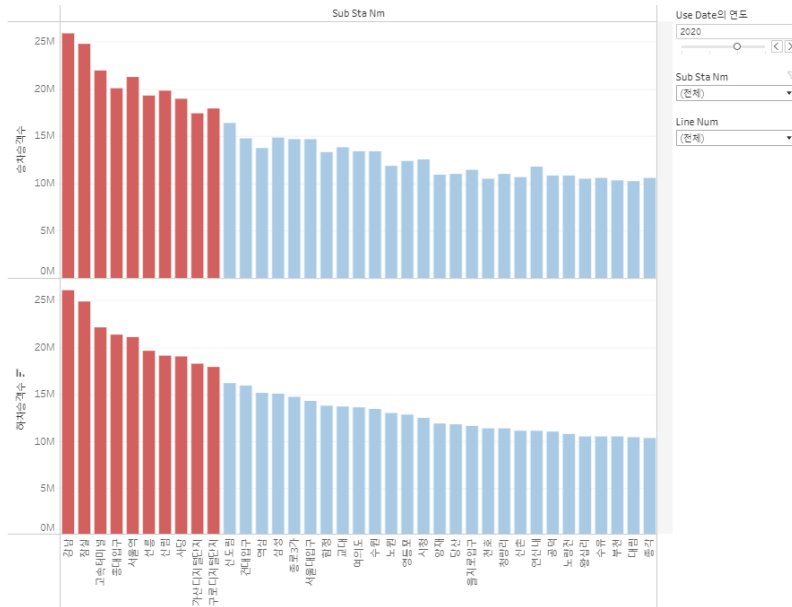
([https://public.tableau.com/app/profile/.19425597/viz/16292715045120/1\\_1?publish=yes](https://public.tableau.com/app/profile/.19425597/viz/16292715045120/1_1?publish=yes))

# 3. 데이터 분석

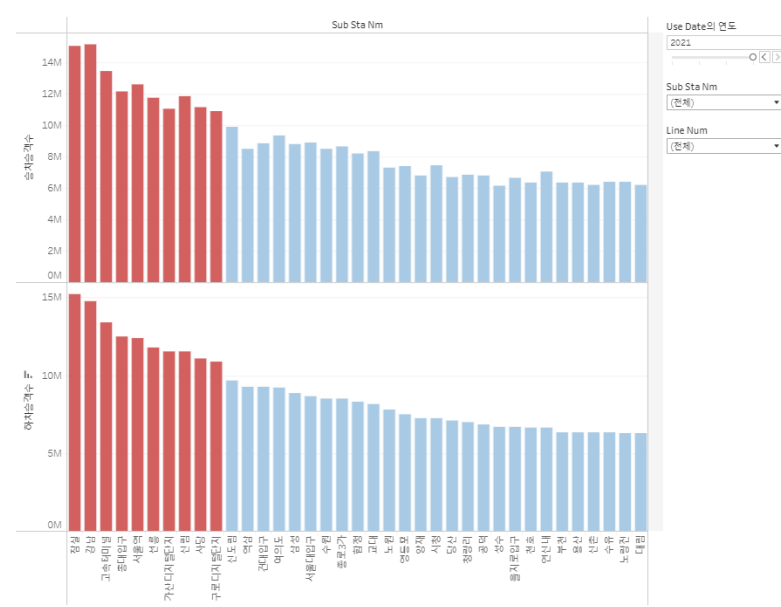
## 1) 2019 ~ 2021년 지하철 역별 총 승객수 변화



코로나 이전  
(2019년)



코로나 이후  
(2020년)

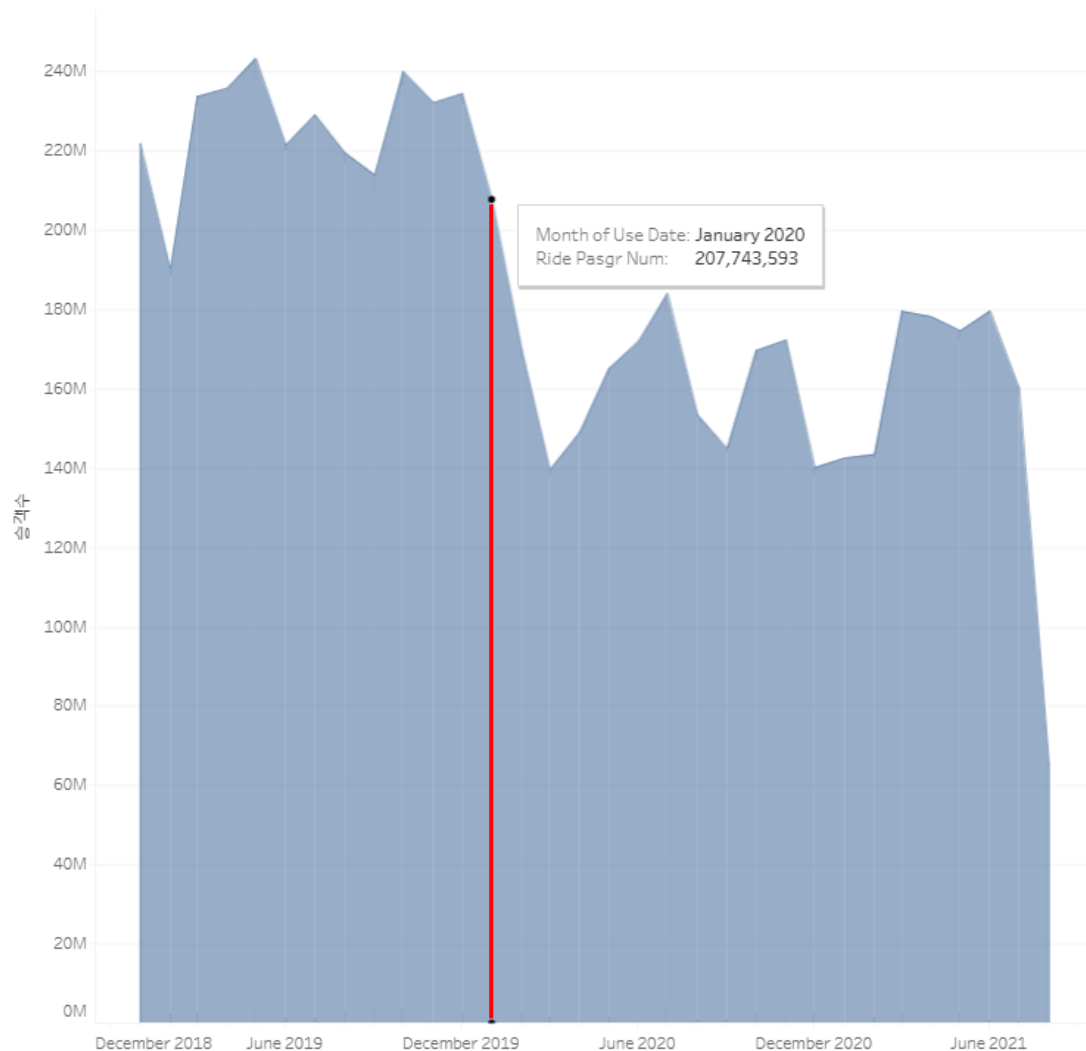


코로나 이후  
(2021년)

코로나 19 발생 이후 매년 지하철 이용승객이 급감

# 3. 데이터 분석

## 2) 2019년 대비 2020년 및 2021년 지하철 이용객 수 변화 분석



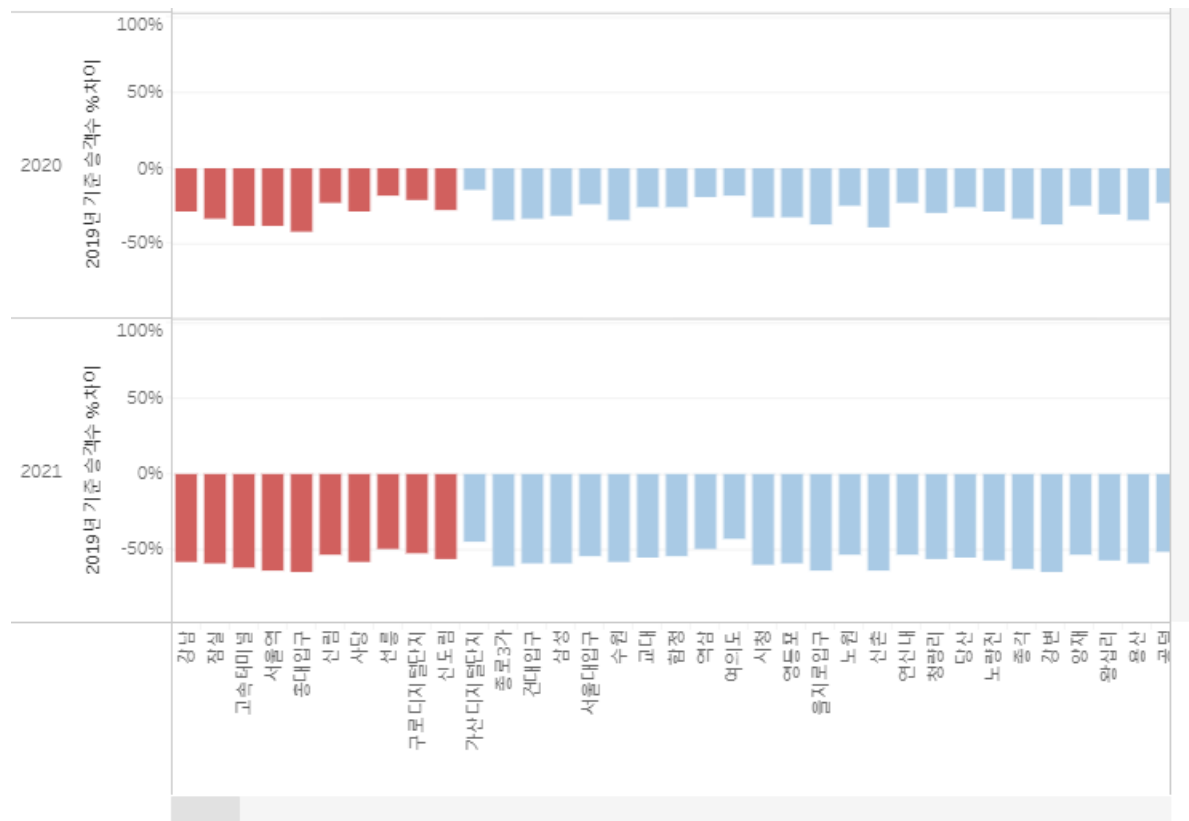
- 2019년 1월 ~ 2021년 8월 월별 지하철 이용객 수
- 2020년 1월을 기점으로 그래프가 급격하게 감소하는 것을 확인할 수 있음

→ 국내 첫 코로나 확진자 발생 이후 지하철 이용객 수가 급격하게 감소했음을 확인



# 3. 데이터 분석

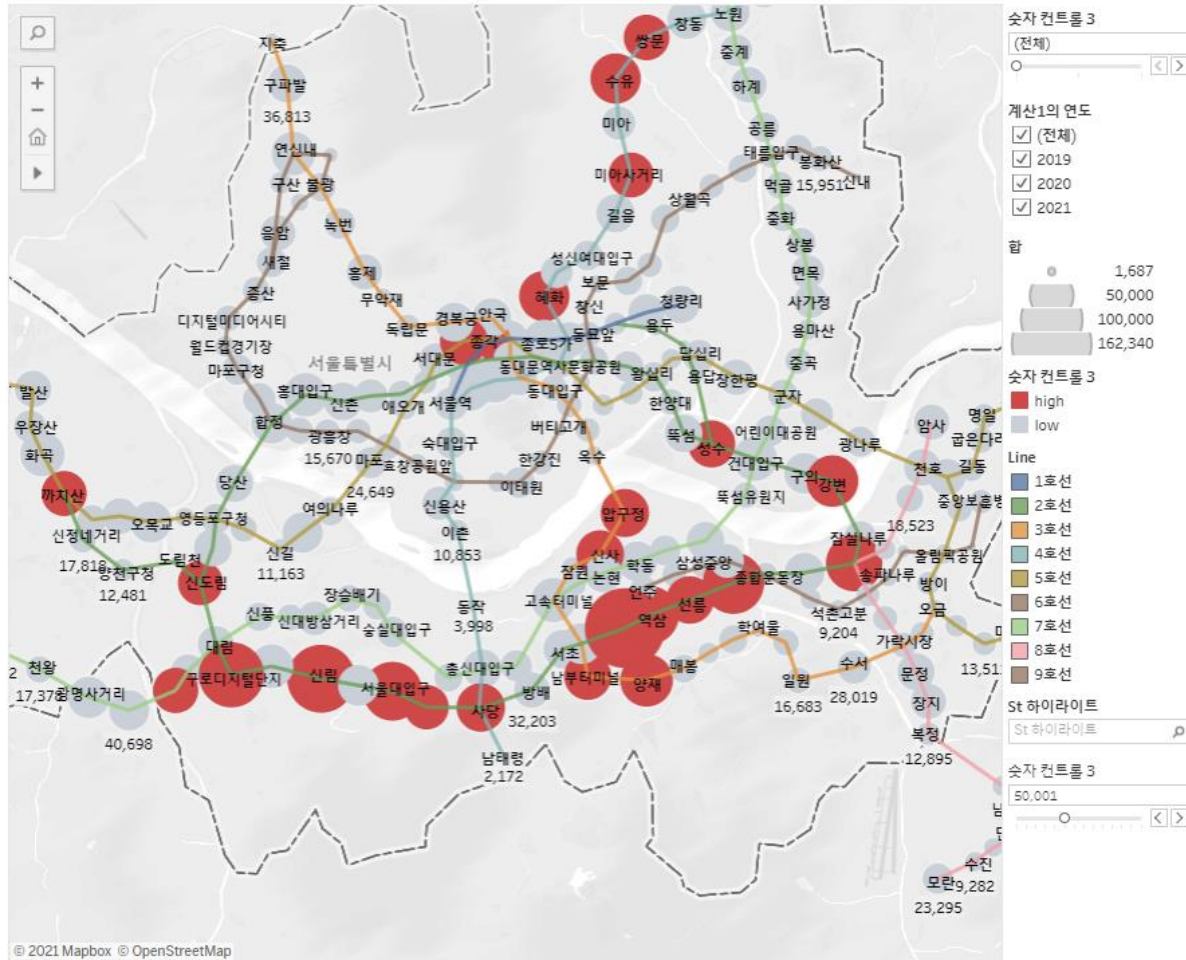
## 3) 2019년 대비 2020년 및 2021년 지하철 이용객 수 변화 비율 분석



- 2019년 대비 2020년 및 2021년 지하철 역별 이용객 수 감소 비율 시각화
- 2019년 대비 2020년
  - 최소 1.1%(마곡 역), 최대 74.8%(경마공원 역) 감소
  - 전반적으로 30~40% 감소한 것을 확인
- 2019년 대비 2021년
  - 최소 0.7%(신내 역), 최대 90.5%(경마공원 역) 감소
  - 전반적으로 40~50% 감소한 것을 확인

# 3. 데이터 분석

## 4) 코로나 이후(2020. 01 ~) 지하철 노선별 일평균 승객수

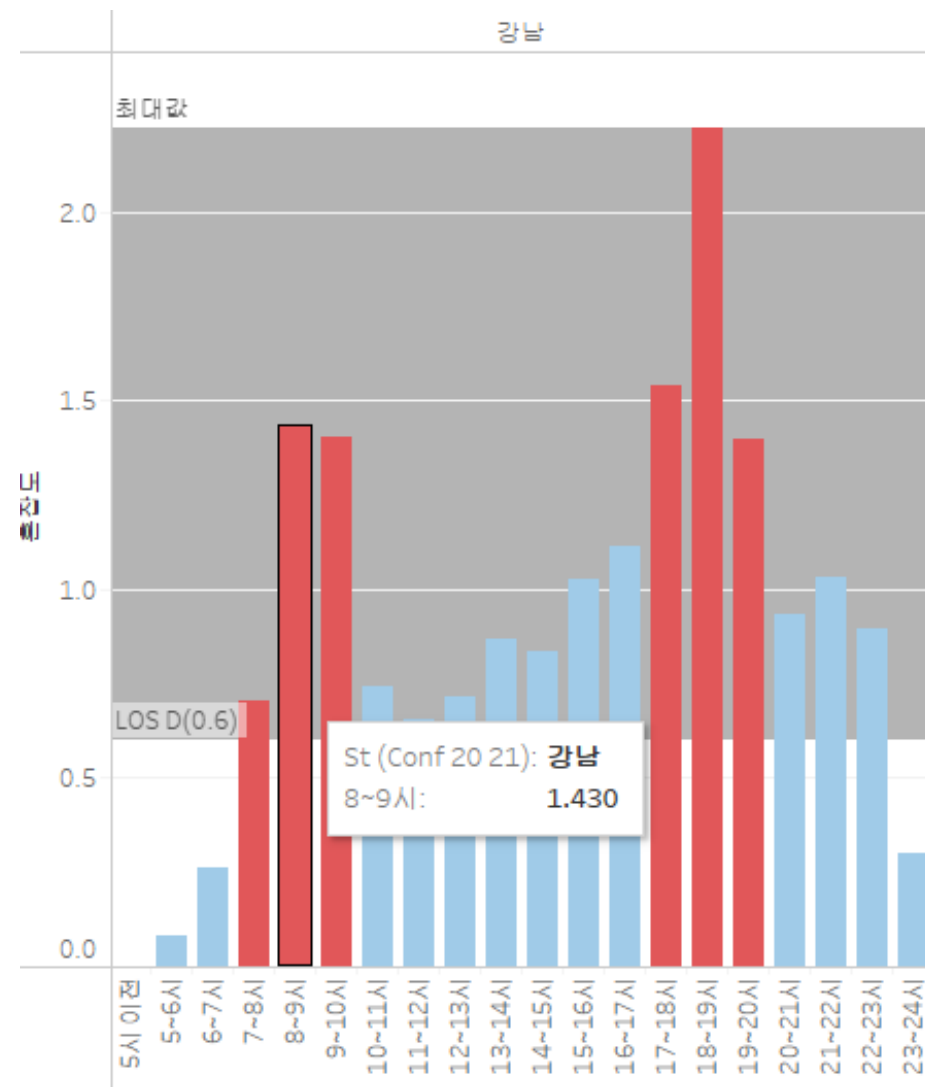


- 코로나 이후(2020. 1 ~ 현재) 일평균 지하철 이용객 수(누적 데이터)
  - 역명과 역별 일평균 승객수 표기
  - 원의 크기는 일평균 승객수 크기에 비례함
  - 원의 색깔은 상대적인 크기를 의미
- : '숫자 컨트롤 3'의 숫자보다 크면 ■ high , 작으면 ■ low (변경가능)

# 3. 데이터 분석

## 5-1) 동적 혼잡도 추정

순위	호선	역명	시간대	혼잡도(%)
1	2호선	강남	18-19시	222
2	2호선	신림	08-09시	167
3	2호선	구로디지털단지	08-09시	164
4	1호선	가산디지털단지	08-09시	150
5	2호선	선릉	18-19시	142
6	2호선	홍대입구	18-19시	141
7	2호선	잠실	18-19시	106
8	2호선	사당	18-19시	77
9	1호선	서울역	18-19시	73
10	3호선	고속버스터미널	18-19시	63



# 3. 데이터 분석

## 5-2) 동적 혼잡도 추정 프로세싱

1단계

- 분석 대상 지역의 대기공간 설정

2단계

- 설정된 대기공간 면적 계산

3단계

- 분석시간대별로 대기공간 내 최대 승객수 측정

4단계

- 설정된 대기공간 면적을 측정된 최대 승객수로 나눠 동적 혼잡도 추정

### 3. 데이터 분석

### 5-3) 승강장 대기공간 면적 계산

## ■ 승강장 실용대기공간 면적

= 승강장 전체면적 - 점유불가면적 - 승객비선회면적

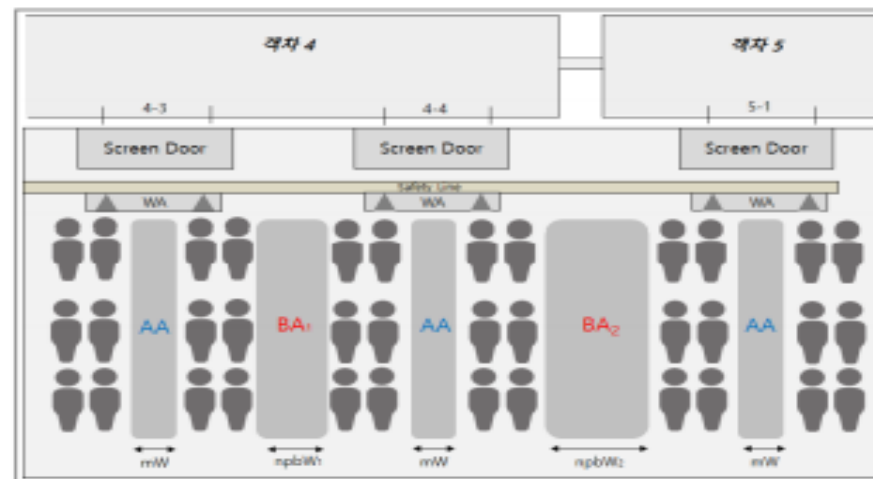
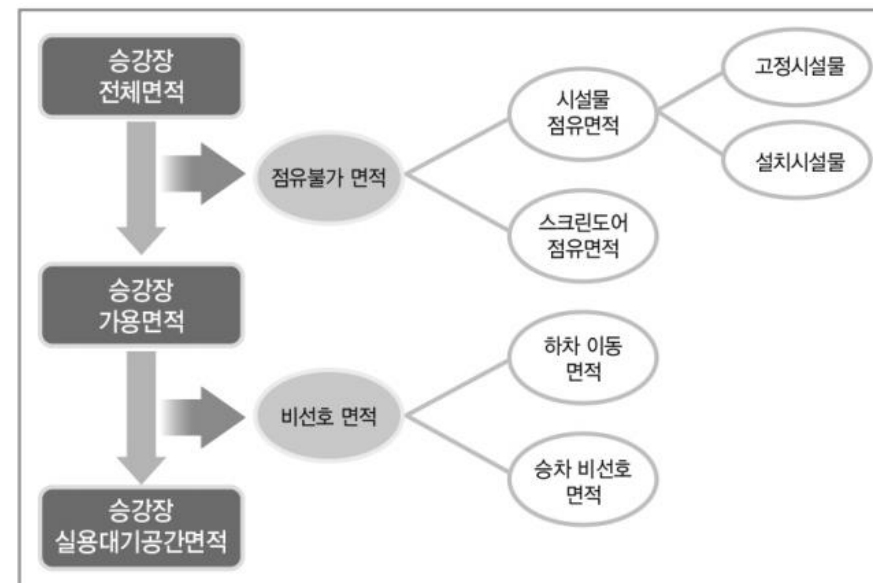
## ■ 점유불가 면적

- 역사운영시설, 편의시설, 소방안전시설, 스크린도어 등
- 승강장 전체 면적 대비 **점유불가 면적은 평균 25%**

## ■ 비선호 면적

- 하차이동면적(AA), 승차비선호면적(BA1, BA2), 출입문 앞
- 승강장 전체 면적 대비 **점유불가 이용자비선호면적 41%**

- 전체적으로 평균 34% 수준을 보이고 있음





# 3. 데이터 분석

## 5-4) 분석시간대별 대기공간 내 최대 승객수 측정

1. 승객수 데이터 기반으로 1분 단위 승객수를 산출하고, 배차간격을 고려하여 승강장 대기 승객수를 산출
2. 승객 통행의 종류를 고려한, 승강장에서의 통행량을 추정
3. 승강장별 실용대기면적을 나누어 동적 혼잡도를 추정

$$C_j^s = \frac{\sum_a X_{ja}^s(t) \cdot a_j^s(t)}{A_j^s}$$

$a$  : 직승직하 승객수(인)

$a_j^s$  : 조절변수(환승승객수, 역사 및 승강장별 영향변수)

$A_j^s$  : s역사 j승강장에서 실용대기면적( $m^2$ )

$X_j^s$  : s역사 j승강장에서 승객(인)

$C_j^s$  : s역사 j승강장에서 승객혼잡도(인/ $m^2$ )

# 3. 데이터 분석

## 5-5) 동적혼잡도 추정

### ■ 서비스수준(Level Of Service/LOS)



- 밀도 =  $\frac{\text{이용고객수}}{\text{실용대기공간면적}}$

- 공간모듈 =  $\frac{\text{실용대기공간면적}}{\text{이용고객수}}$

### ■ 한국인 체형을 적용하여 첨두시간대 승강장 서비스수준은 D로 설계하도록 제시

### ■ 따라서 서비스수준 D 이하는 혼잡한 역사로 분류

서비스 수 준	공간모듈 (m²/인)	평균간격 (cm)	밀도 (인/m²)	보 행 상 태
A	1.3 이상	120 이상	0.8 이하	자유흐름의 영역
B	1.0-1.3	105-120	1.0-0.8	타인을 무리없이 통과 가능
C	0.7-1.0	90-105	1.4-1.0	타인 통과시 불편을 끼침
D	0.3-0.7	60-90	3.3-1.4	타인과의 접촉없이 대기 가능
E	0.2-0.3	60이하	5.0-3.3	타인과의 접촉없이 대기 불가능
F	0.2 이하	packed state	5.0이상	타인과 밀착, 심리적 불쾌상태

D	비접촉영역의 한계 (직경 90 cm)	
E ~ F	접촉영역의 한계 (직경 60 cm)	



---

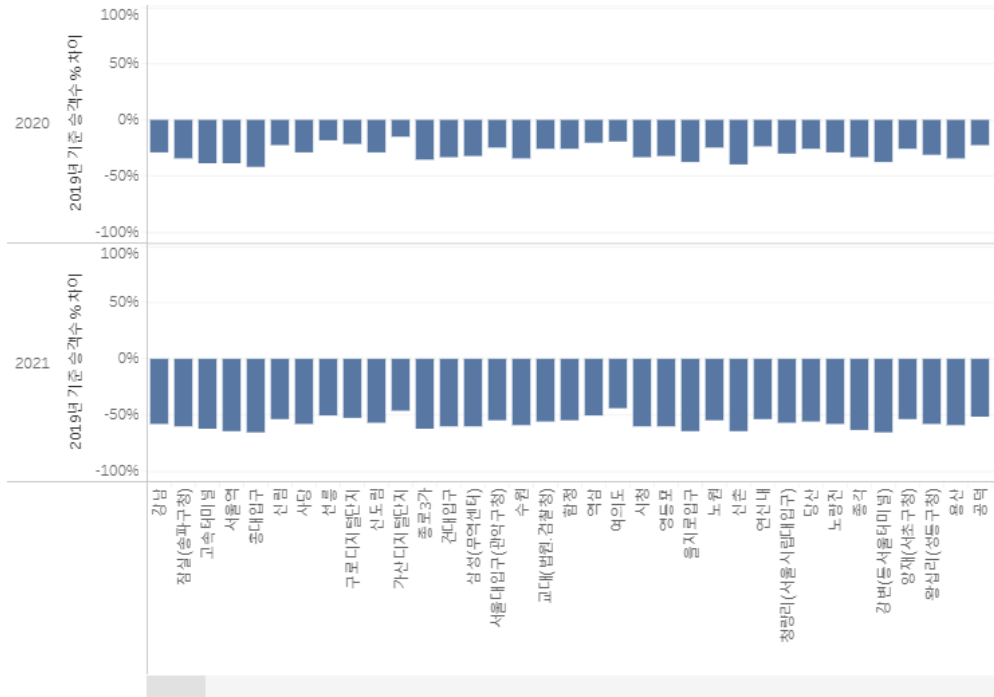
## Part 4.

# 기대 효과

- 분석을 통한 인사이트 도출
- 향후 개선 사항 및 기대 효과

# 4. 기대 효과

## 1) 분석을 통한 인사이트 도출



순위	호선	역명	시간대	혼잡도(%)
1	2호선	강남	18-19시	222
2	2호선	신림	08-09시	167
3	2호선	구로디지털단지	08-09시	164
4	1호선	가산디지털단지	08-09시	150
5	2호선	선릉	18-19시	142
6	2호선	홍대입구	18-19시	141
7	2호선	잠실	18-19시	106
8	2호선	사당	18-19시	77
9	1호선	서울역	18-19시	73
10	3호선	고속버스터미널	18-19시	63



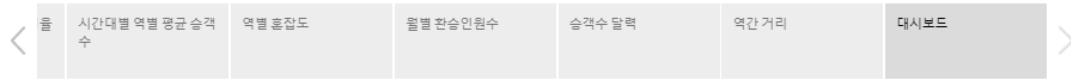
코로나 19로 인해 지하철 총 이용객 수는 급감함

하지만 여전히 출퇴근 시간대에는 특정 역들에서 혼잡도가 매우 높게 나타나는 것을 확인

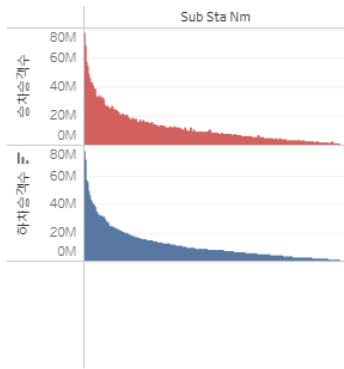
# 4. 기대 효과

## 2) 향후 개선사항 및 기대효과

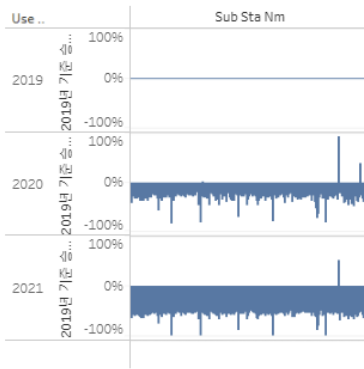
서울 지하철 비교 분석



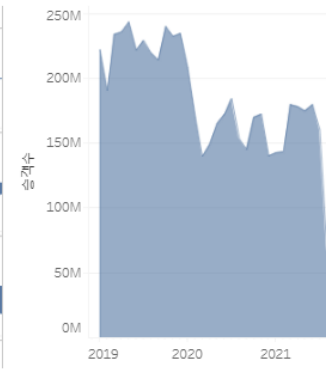
역별 전체 승객수



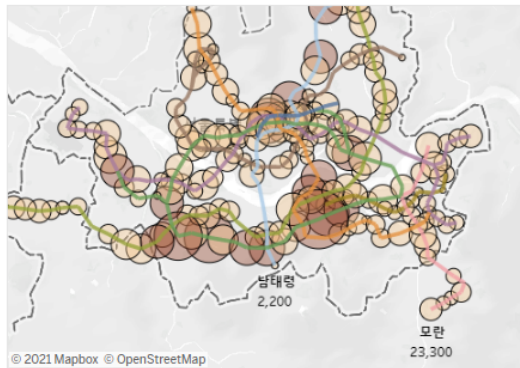
연도별 승객수 변화 비율



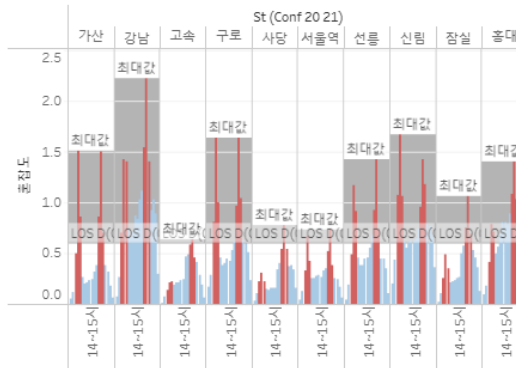
연도별 승객수 변화



지하철 노선별 승객수



역별 혼잡도



- 시각화된 데이터를 바탕으로 방역 강화, 지하철 서비스 혹은 정책 개선 가능

ex 1) 역별 혼잡도 데이터를 바탕으로 혼잡도가 높은 역, 시간대에 대해 방역 강화

ex 2) 일평균 지하철 노선별 승객수 데이터를 바탕으로 새로운 서비스나 개선 사항을 역별로 선택적으로 적용

- 교통 서비스 성과 증대, 이동성 위험의 지속성 완화 등에 기여할 수 있을 것으로 예상



# 4. 기대 효과

## 2) 향후 개선사항 및 기대효과

	LINE_NUM	SUB_STA_NM	RIDE_PASGR_NUM	ALIGHT_PASGR_NUM	USE_DATE
▶	1호선	서울역	25243	20057	2021-08-16
	경춘선	평내호평	3936	3851	2021-08-16
	1호선	동묘앞	6964	7314	2021-08-16
	경춘선	천마산	1260	1201	2021-08-16
	1호선	시청	6639	6200	2021-08-16
	경춘선	마석	2850	2812	2021-08-16
	1호선	종각	11972	10941	2021-08-16
	경춘선	대성리	586	450	2021-08-16
	1호선	종로3가	18568	17283	2021-08-16
	경춘선	척평	1714	1378	2021-08-16

< 8월 20일 오전 10시 기준 최신 지하철 승하차인원 데이터 >



< SKT, 서울 지하철 2호선 칸별 '혼잡도' 실시간 제공 >

- 현재 Open API로부터 수집하고 있는 데이터는 일일 승하차 총인원 데이터
- 실시간 승하차인원 데이터가 수집 가능할 경우, 역별, 칸별 혼잡도를 실시간으로 제공 가능
- 상대적으로 혼잡도가 낮은 지하철 칸을 이용하거나 다른 대중교통 수단을 이용하도록 유도하는 등 지하철 혼잡도를 완화하는데 기여할 수 있음



---

## Part 5.

# 개발 후기 및 느낀점

- 개발 후기 및 느낀점
- 참고 문헌 및 보도자료

# 5. 개발 후기 및 느낀점

## ✓ 허재혁 님

해당 프로젝트를 통해 다양한 툴을 AWS 서버와 연동하여 데이터 저장 시스템을 구축하고, Tableau를 활용하여 많은 양의 데이터를 시각화하는 경험을 할 수 있는 좋은 프로젝트였습니다.

## ✓ 김정명 님

처음 진행을 진행하였을 때 많이 알지 못한 상태에서 시작해서 걱정부터 앞섰습니다. 하지만 프로젝트를 진행하면서 분업도 너무 잘 되었고 팀원들이 맡은 업무도 너무 열심히 해주어서 잘 마무리할 수 있었던 것 같습니다. 진행하면서 많은 부분을 배웠고 재밌는 시간이었습니다.

## ✓ 박성하 님

팀원들의 도움으로 데이터 파이프라인의 전반적인 흐름을 이해할 수 있었으며, tableau를 이용해 여러 데이터를 다루고 시각화까지 이어질 수 있어 많은 것을 배우게 되었습니다.

## ✓ 정길종 님

우선 팀원들 개발에 대한 역량이 모두 높으셔서 도움을 많이 받았고, 여러가지로 공부할 수 있었던 기회였습니다. 그리고 프로젝트를 통해서 AWS로 클라우드 개발 환경을 경험한 것, 리눅스 환경에서 데이터 파이프라인을 구축을 통해서 수집 및 저장까지의 프로세스를 이해 할 수 있었습니다.

# 수집 데이터 및 참고 자료

## ■ 수집 데이터

- 서울시 지하철 호선별 역별 시간대별 승하차 인원 정보(10`~21`)
- 역별 혼잡도(15,17,19년도)
- 서울교통공사 2019년 환승역거리 및 소요시간 정보
- 서울교통공사 역간거리
- 서울교통공사 역사면적정보(19`)
- 서울교통공사 월별 환승유입인원(21`)

## ■ 참고 자료 및 논문

- 도시철도 정거장 및 환승. 편의시설 설계(국토교통부, 2018. 3. 28)
- 신성일, 이상준, & 이창훈. (2019). 스마트카드자료를 활용한 지하철 승강장 동적 혼잡도 분석모형
- 최진경, 이호, & 유봉석. (2014). 대중교통카드 자료를 활용한 실시간 도시철도 승강장 혼잡도 분석 알고리즘 개발
- 이호, 장기백, & 유봉석. (2016). 도시철도 역사 승강장 실용대기공간면적 산정 연구
- 홍유정, 한채연. (2020). 코로나 19 확산 이후 서울시 지하철 이용 변화분석
- 신희강 기자, "SKT, 서울 지하철 2호선 칸별 '혼잡도' 실시간 제공" (뉴데일리 경제, 2021. 8. 18)



**Q & A**

**감사합니다.**