

코로나19 프로젝트

프롬프트 작성

▼ 대충 프롬프트

젼미니에게

코로나 감염자의 시작 부터 세계적으로 감염 확산되는
과정을 보여 주는 대쉬보드를 만들려고해
좀 아웃을 하면 세계지도가 보여지고 줌인을 하면 각국의 나라가 보여 진다.
그리고사람들이 이동하는 모습이 보여 져
나라를 이동하면 비행디로 이동이 된다.
이런 대쉬 보드를 만들려고 하는데
여기서
통계를 찾아야하는데 너가 찾아 봐.
그리고 위의 글을 전문가 프롬프트로 다시 작성해 줘

AI의견

코로나19의 초기 발생부터 전 세계적 확산을 시각화하는 대시보드 기획은 데이터의 흐름을 직관적으로 보여주기에 아주 훌륭한 아이디어입니다. 특히 **항공기 이동 경로**를 포함하는 것은 바이러스의 전파 원인을 설명하는 데 핵심적인 요소입니다.

요청하신 **통계 데이터 출처**와 이를 기반으로 프로젝트를 구체화한 **전문가 프롬프트**를 정리해 드립니다.

▼ 1단계: 프로젝트 구현을 위한 주요 데이터 출처 수집

대시보드 구현에 필요한 데이터를 세 가지 카테고리로 나누어 찾았습니다.

① 전 세계 코로나19 확진 및 전파 통계

- **Our World in Data (OWID):** 국가별 일일 확진자, 사망자, 백신 접종률 등 가장 정제된 시계열 데이터를 제공합니다. (GitHub/CSV/JSON 제공)

- **WHO COVID-19 Dashboard:** 세계보건기구 공식 통계로, 공신력 있는 국가별 데이터를 API로 연동할 수 있습니다.
- **Johns Hopkins University (JHU) Archive:** 팬데믹 초기(2020-2023)의 상세한 전파 기록을 담고 있는 가장 유명한 데이터셋입니다.

② 국가 간 이동 및 항공기 궤적 데이터

- **OpenSky Network:** 전 세계 항공기 위치 데이터를 실시간/과거 기록으로 제공합니다. 특정 시점의 비행기 이동 패턴을 시각화할 때 필수적입니다.
- **ICAO (국제민간항공기구):** 팬데믹 기간 중 항공 노선 변화 및 이동량 감소에 대한 경제/물리적 통계를 제공합니다.

③ 인구 이동성 데이터 (줌인 시 유용)

- **Google Community Mobility Reports:** 사람들이 소매점, 공원, 주거지 등으로 어떻게 이동했는지에 대한 변화율을 국가/도시별로 제공합니다. (2022년 말까지의 데이터 제공)

▼ 2단계: 구성 프롬프트

이 프롬프트는 개발자나 AI(GPT-4, Claude 등)에게 입력하여 **기술적인 요구사항을 명확히 전달**하기 위한 용도입니다.

[Dashboard Project: Global COVID-19 Spatio-Temporal Transmission Visualizer]

1. Project Overview

팬데믹 초기 단계부터 현재까지의 코로나19 확산 과정을 시계열(Time-lapse)로 시각화하는 인터랙티브 대시보드를 구축하고자 함. 단순한 수치 나열을 넘어, 국가 간 물리적 이동(항공 경로)이 바이러스 전파에 미친 영향을 직관적으로 보여주는 것이 목적임.

2. Key Visualization Features

- **Multi-level Zoom Interface:** Mapbox나 Deck.gl을 활용하여 '세계 지도(Global View)'에서 '국가/도시별 상세 지도(Local View)'로 매끄럽게 전환되는 인터페이스 구현.
- **Dynamic Mobility Animation:** 국가 간 인구 이동을 점과 선의 흐름으로 표현. 특히 국가 이동 시 실제 항공기 항로를 기반으로 한 Arc(호) 형태의 애니메이션 효과 적용.

- **Epidemiological Overlay:** 확진자 발생 밀도를 히트맵(Heatmap)이나 버블 차트(Bubble Chart)로 레이어링하여, 항공기 이동 후 해당 지역의 확진자 증가 양상을 시각적으로 동기화.

3. Data Requirements

- **Epidemic Data:** Our World in Data(OWID) 또는 JHU의 API를 사용하여 국가별 일일 신규 확진자 및 누적 통계 확보.
- **Flight Path Data:** OpenSky Network의 역사적 데이터를 활용하여 주요 허브 공항 간의 비행 노선 시각화.
- **Timeline Control:** 하단 타임 슬라이더를 통해 2019년 12월부터 현재까지의 확산 속도와 경로를 자유롭게 탐색 가능하도록 설정.

4. Technical Stack Recommendation

- **Frontend:** React.js / Vue.js
- **Map Engine:** Mapbox GL JS 또는 Leaflet
- **Data Visualization:** Deck.gl (대량의 아크 및 포인트 애니메이션 처리에 최적화)
- **Backend:** Python (FastAPI/Flask) 기반의 데이터 전처리 파이프라인

▼ 바이브 코딩

클라우드 AI 에게

코드제작 맞기기

오류 발생 지구가 안 그려짐

계속 수정

코드 꼬임 발생

그록 AI 에게 전달

오류 수정 80%

다시 Gemini 에게 전달

오류 수정 99%

배포

<https://app.netlify.com/projects/magical-kulfi-3c300e/overview>

<https://magical-kulfi-3c300e.netlify.app/>

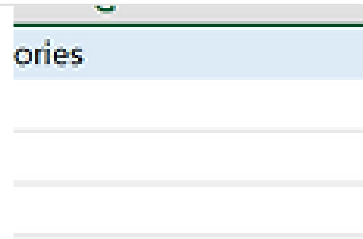
<https://magical-kulfi-3c300e.netlify.app/>

▼ 데이터 찾기

코로나19(Corona19, COVID19) 엑셀 데이터 얻기/가공하기(1/2)

이 글은, 전 세계 코로나19 신규 확진자 수, 누적 확진자 수, 신규 사망자 수 등의 데이터를 얻고자 할 때 유용한 지식입니다. 코로나 데이터를 얻는 여러 가지 방법이 있는데, 여기서 소개할 것은, 데이터 전체를 엑셀로 받는 법

 <https://infograph.tistory.com/87>



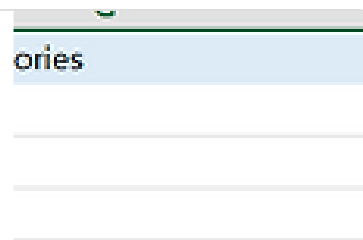
data.europa.eu

<https://data.europa.eu/data/datasets/covid-19-coronavirus-data?locale=en>

코로나19(Corona19, COVID19) 엑셀 데이터 얻기/가공하기(1/2)

이 글은, 전 세계 코로나19 신규 확진자 수, 누적 확진자 수, 신규 사망자 수 등의 데이터를 얻고자 할 때 유용한 지식입니다. 코로나 데이터를 얻는 여러 가지 방법이 있는데, 여기서 소개할 것은, 데이터 전체를 엑셀로 받는 법

 <https://infograph.tistory.com/87>



The official portal for European data | data.europa.eu

Discover the single point of access to open data from European countries, EU institutions, agencies and bodies and other European countries.

 <https://data.europa.eu/en>

▼ 발표는 스토리를 담아서

[3H칼럼] 왜 우리는 2년간 코로나 바이러스와 동거하고 있나
[3H칼럼] 왜 우리는 2년간 코로나 바이러스와 동거하고 있나 / 환자를
가장 먼저 생각하는 건강전문지

■ <https://www.hnews.kr/news/view.php?no=58333>



▼ 참고자료들

[https://raw.githubusercontent.com/owid/covid-19-data/refs/heads/master/public/data/cases_deaths/COVID-19 Cases and deaths - WHO.csv](https://raw.githubusercontent.com/owid/covid-19-data/refs/heads/master/public/data/cases_deaths/COVID-19%20Cases%20and%20deaths%20-%20WHO.csv)

아래는 사용자가 주신 OWID의 WHO COVID-19 데이터셋을
파이썬 + pandas로 불러와서 정리하고, 분석하기 좋게 만드는 방법을 단계별로 정리한 코
드

1. 기본 불러오기 & 기본 정리

```
import pandas as pd

# 데이터 URL
url = "https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/cases_deaths/COVID-19%20Cases%20and%20deaths%20-%20WHO.csv"

# 데이터 불러오기
df = pd.read_csv(url)

# 기본 정보 확인
print("데이터 크기:", df.shape)
print("\n컬럼 목록:")
print(df.columns.tolist())
print("\n최신 날짜:", df['date'].max())
print("국가/지역 수:", df['location'].nunique())

# 결측값 비율 확인
print("\n결측값 비율:")
print(df.isna().mean().sort_values(ascending=False).head(10))
```

```

# 날짜를 datetime으로 변환
df['date'] = pd.to_datetime(df['date'])

# 불필요하거나 거의 빈 컬럼 제거 (필요에 따라 조정)
columns_to_drop = [col for col in df.columns if 'smoothed'
not in col and 'per_million' not in col and col not in [
    'location', 'iso_code', 'date', 'new_cases', 'new_deaths',
    'total_cases', 'total_deaths'
]]

# 실제로 많이 사용하는 핵심 컬럼만 남기기 (선택적)
useful_columns = [
    'location', 'iso_code', 'date',
    'new_cases', 'new_deaths',
    'total_cases', 'total_deaths',
    'new_cases_smoothed', 'new_deaths_smoothed',
    'weekly_new_cases_per_million', 'weekly_new_deaths_per_
million'
]

df = df[useful_columns].copy()

print("\n정리된 데이터 미리보기:")
print(df.tail(8))

```

2. 국가별 최신 데이터 추출

```

# 가장 최근 날짜의 데이터만 추출
latest_date = df['date'].max()
latest = df[df['date'] == latest_date].copy()

# 누적 확진/사망 기준 상위 15개국
print("\n누적 확진 상위 15개국 (최신 기준)")
print(latest.sort_values('total_cases', ascending=False)
[['location', 'total_cases', 'total_deaths']].head(15))

# 인구당 주간 신규 확진 상위 15개국
print("\n인구 100만 명당 주간 신규 확진 상위 15개국")

```

```
print(latest.sort_values('weekly_new_cases_per_million', ascending=False)[
    ['location', 'weekly_new_cases_per_million', 'new_cases_smoothed']
].head(15))
```

3. 한국(South Korea) 데이터만 추출하여 정리

```
# 한국 데이터만 필터링
korea = df[df['location'] == 'South Korea'].copy()

# 최근 10주 데이터 보기
print("\n한국 최근 10주 데이터:")
print(korea.tail(10)[['date', 'new_cases', 'new_cases_smoothed',
                    'new_deaths', 'weekly_new_cases_per_million']])

# 간단한 통계
print("\n한국 전체 기간 통계:")
print(korea[['new_cases', 'new_deaths', 'total_cases', 'total_deaths']].describe())
```

4. 월별/연도별 집계 (예시)

```
# 연도-월 컬럼 추가
korea['year_month'] = korea['date'].dt.to_period('M')

# 월별 집계
monthly = korea.groupby('year_month').agg({
    'new_cases': 'sum',
    'new_deaths': 'sum',
    'new_cases_smoothed': 'mean'
}).reset_index()

print("\n한국 월별 신규 확진/사망 집계:")
print(monthly.tail(12))
```

```
# 시각화하고 싶다면 (matplotlib)
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(monthly['year_month'].astype(str), monthly['new_cases'], label='월간 신규 확진')
plt.plot(monthly['year_month'].astype(str), monthly['new_cases_smoothed'] * 4,
          label='주간 평균 × 4 (근사 월간)', linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
plt.title('한국 월별 COVID-19 신규 확진 추이')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

5. 전체 국가 중 원하는 국가들만 추출하는 함수

```
def get_country_data(df, country_names):
    """
    원하는 국가들의 데이터만 반환
    예: get_country_data(df, ['South Korea', 'Japan', 'United States'])
    """
    return df[df['location'].isin(country_names)].copy()

# 사용 예시
countries = ['South Korea', 'Japan', 'United States', 'Germany', 'United Kingdom']
selected = get_country_data(df, countries)

# 최근 데이터 비교
latest_selected = selected[selected['date'] == selected['date'].max()]
print("\n선택한 국가들 최근 주간 데이터 비교:")
print(latest_selected[['location', 'date', 'new_cases_smoothed'],
```



```
        'weekly_new_cases_per_million']]  
t_values('weekly_new_cases_per_million', ascending=False))
```

6. 저장하고 싶을 때

```
# csv로 저장  
df.to_csv('covid_who_cleaned.csv', index=False)  
korea.to_csv('covid_korea.csv', index=False)  
monthly.to_csv('covid_korea_monthly.csv', index=False)
```