[프로그래머스 데이터 엔지니어링 데브코스 1기] 2023.07.07

2차 프로젝트

2팀 1조

공공데이터 API를 이용한 노인일자리 분석

김상희, 김혜민, 남윤아, 이성희, 이하윤

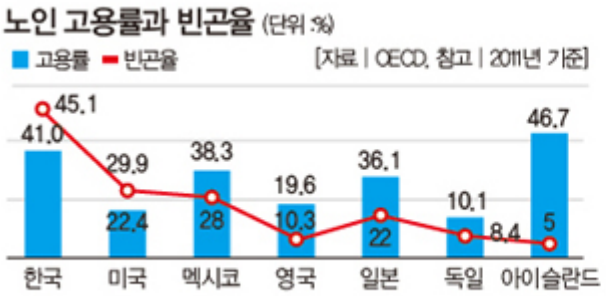
목차

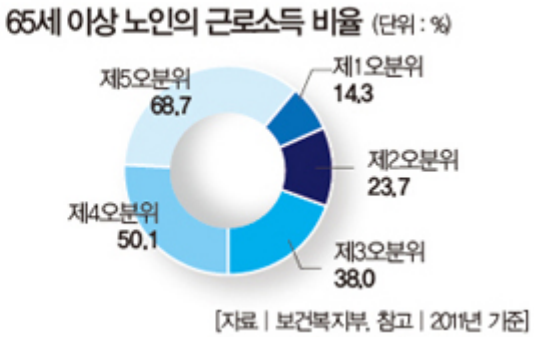
1. 프로젝트 개요

노인일자리 사업은 65세 이상 노인들에게 경제적 자립과 사회 참여를 목적으로 정부에서 시행하고 있는 복지사업입니다. 최근 3개년 동안의 노인일자리 공고와 사업 정보를 수집하여 노인일자리 채용 현황과 고용 지속성을 조사합니다.

2. 프로젝트 주제 선정 이유

노인일자리 사업은 경제적 자립과 사회 참여를 촉진하는데 중요한 역할을 합니다. 그러나 사업의 실제 현황과 채용 동향에 대한 정확한 정보 부족으로 인해 정책의 효과성을 평가하기 어렵습니다. 이 프로젝트는 노인일자리의 공고와 사업 정보를 분석하여 노인일자리 구인 동향을 파악하고, 정부의 노인 복지 정책의 효과성을 평가할 수 있습니다.





3. 기대효과

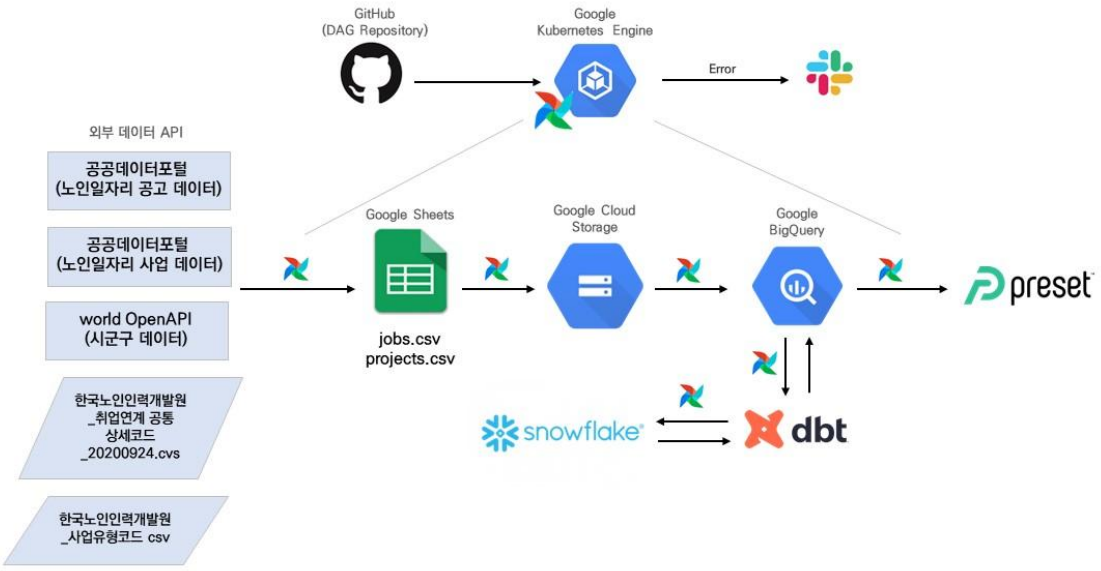
1. 노인일자리에 대한 상세한 이해를 도모할 수 있습니다.
2. 노인일자리 정책의 성과와 한계를 파악하고 개선할 수 있는 방안을 모색할 수 있습니다.

4. 활용 기술 및 프레임워크

A. 개발 환경

Field	Stack
사용 언어	python
데이터 적재	bigquery, Google Cloud Storage, google sheets
ETL & ELT	airflow, DBT
infra	Google Cloud Platform
CI/CD	Git Action
Dashboard	Superset
협업 도구	github, slack, notion

B. 프로젝트 아키텍처



C. ERD

〈Jobs(공고) 테이블〉

jobs	
jobId	**varchar(20)**
jobcls	varchar(50)
jobclsNm	varchar(50)
acptMthd	varchar(2000)
deadline	varchar(50)
emplymShp	char(6)
emplymShpNm	varchar(6)
startDd	char(10)
oranNm	varchar(50)
organYn	char(1)
recrtTitle	varchar(1000)
stmId	char(1)
stmNm	varchar(50)
endDd	varchar(10)
workPlc	varchar(50)
acptMthdCd	varchar(6)
age	varchar(2)
ageYn	char(1)
clerk	varchar(50)
clerkContt	varchar(15)
cllfPrnnum	varchar(4)
createDt	varchar(100)
detCnts	varchar(4000)
etcltm	varchar(4000)
homepage	varchar(100)
plDetAddr	varchar(200)
plbizNm	varchar(1000)
updDt	varchar(100)
sysCreatedAt	timestamp_ntz

〈Projects(사업) 테이블〉

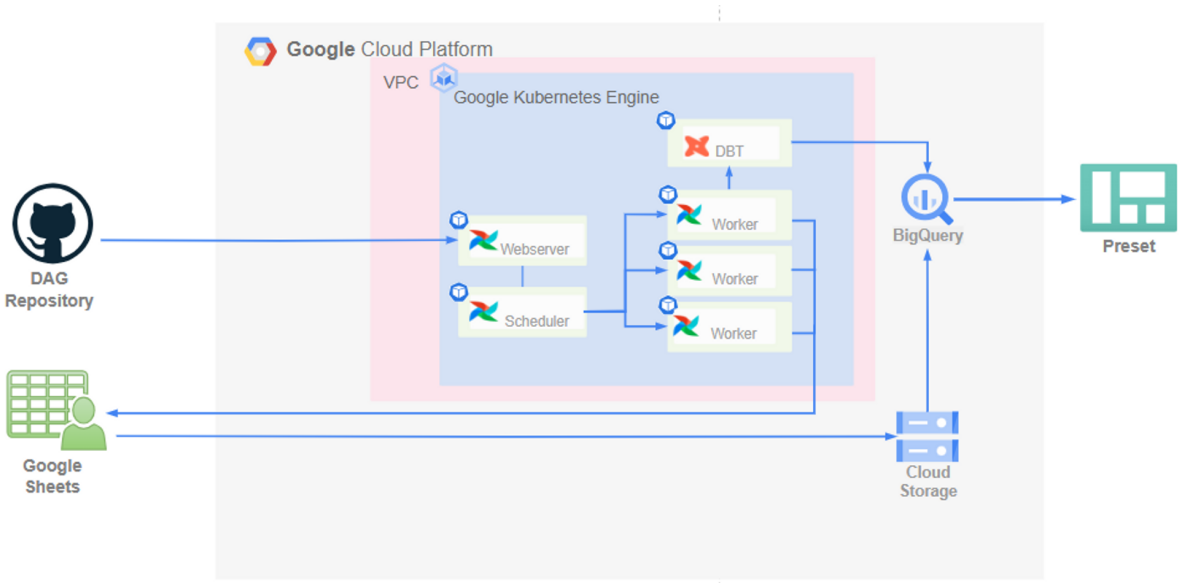
projects	
projNo	**char(20)**
projType	varchar(30)
projPlanChangeNo	int
projYear	char(4)
contProjYn	char(1)
contProjStartYear	char(4)
projTypeCd	char(10)
projTypeNm	varchar(100)
nonBudgYn	char(1)
specProjCd	varchar(10)
projNm	varchar(100)
admProvNm	varchar(10)
admDistCd	varchar(10)
admDistNm	varchar(50)
institutionId	varchar(10)
projStartDd	char(10)
projEndDd	char(10)
planStatusCd	varchar(20)
targetEmployment	int
firstlAttachment	varchar(100)
recentApprovalAttachment	varchar(100)
delYn	char(1)
sysCreatedAt	timestamp_ntz

5. 프로젝트 세부 결과

A. GCP 인프라 구축

파이프라인 구축 및 실시간 처리가 가능하도록 GCP 및 Airflow 를 도입했습니다.

3차 프로젝트의 인프라 구조도는 다음과 같습니다.



- Google Kubernetes Engine: 클라우드 기반 Kubernetes 서비스
    - Airflow : GKE 위에 Airflow가 동작함
- Github : DAG 저장소
- Google Sheets : 공공데이터 API 데이터가 1차 적재
- Google Cloud Storage : Google Sheets 데이터를 가져와 2차 적재
- BigQuery : Raw data와 Analytics, Adhoc 스키마 저장소
- Preset : Bigquery에서 데이터를 가져와 시각화

1) Helm 차트를 활용하여 GKE에 Airflow 구축

Apache Airflow Helm Chart를 사용하여 GKE에 Airflow를 배포했습니다. Airflow는 GKE 클러스터 내에서 실행되며 웹 서버, 스케줄러, 워커, PostgreSQL 등의 컴포넌트가 Pod로 실행됩니다. Airflow를 사용하여 ETL 및 ELT 등 작업 스케줄링 및 워크플로우 관리를 가능하게 합니다.

[환경 설정]

(1) Kubernetes 클러스터 생성

- Google Kubernetes Engine(GKE)에서 새로운 Kubernetes 클러스터를 생성함. 클러스터는 작업을 실행할 환경을 제공함
- 리전은 리전별 가격 비용을 통해 us-central1로 선정함
- GKE 클러스터는 Standard형으로 생성함

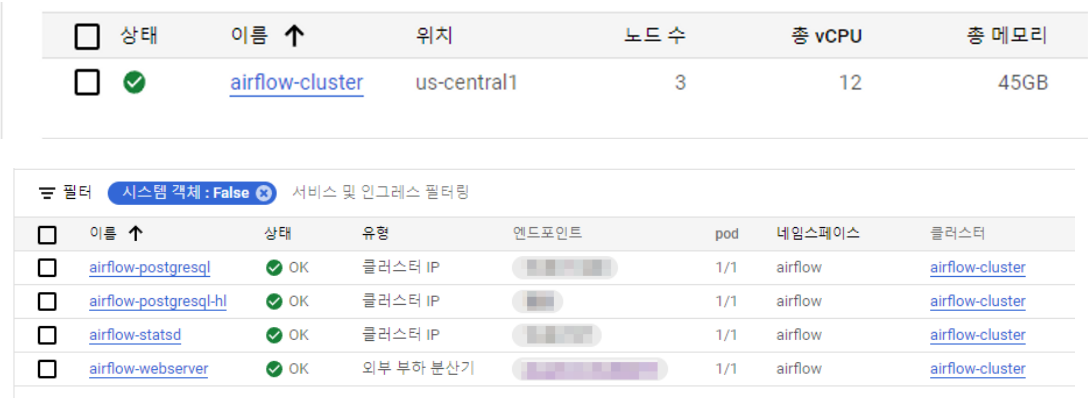
(2) 'airflow' 네임스페이스 생성

- 리소스의 범위를 제한하기 위해 'airflow' 네임스페이스 생성

(3) Airflow Helm Chart 설치

- Helm 차트 저장소에 Apache Airflow 공식 Helm 차트를 추가하고 values.yaml 파일을 사용하여 Airflow Helm Chart를 수정 및 배포함
- 웹 서버, 스케줄러, 워커, 메타데이터 데이터베이스 PostgreSQL 등이 Pod 형태로 생성됨

*Helm은 Kubernetes 애플리케이션을 손쉽게 배포하고 관리하기 위한 패키지 관리 도구



상태	이름	위치	노드 수	총 vCPU	총 메모리
✓	airflow-cluster	us-central1	3	12	45GB

필터 | 시스템 객체 : False | 서비스 및 인그레스 필터링

이름	상태	유형	엔드포인트	pod	네임스페이스	클러스터
airflow-postgresql	OK	클러스터 IP	[redacted]	1/1	airflow	airflow-cluster
airflow-postgresql-hl	OK	클러스터 IP	[redacted]	1/1	airflow	airflow-cluster
airflow-statsd	OK	클러스터 IP	[redacted]	1/1	airflow	airflow-cluster
airflow-webserver	OK	외부 부하 분산기	[redacted]	1/1	airflow	airflow-cluster

[values.yaml 파일 사용자 정의]

(1) KubernetesExecutor 사용

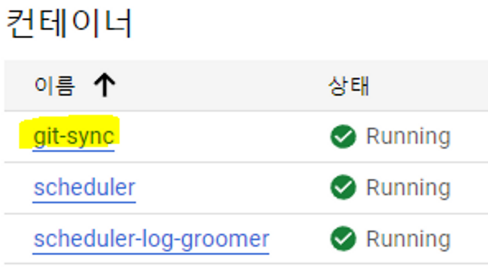
a) **확장성**: Airflow 작업마다 Pod를 동적으로 생성하여 스케일링하고 자원을 효율적으로 사용할 수 있음
b) **격리성**: 각각의 Airflow 작업이 Pod에서 실행되므로 격리성을 보장하며 작업 간의 영향을 최소화함
c) **쉬운 배포**: Kubernetes는 컨테이너화된 작업을 관리하며, 작업의 배포 및 롤백, 로깅, 모니터링 등을 지원함. 또한 Helm등을 이용하여 작업 관련 리소스를 쉽게 패키징하고 배포 가능함

(2) Git Sync DAGs 배포하기

- Git Sync는 Git 저장소에서 Airflow 배포로, DAG(Directed Acyclic Graph)를 자동으로 동기화할 수 있는 Airflow의 기능을 이용하여 DAG 저장소인 Git Repository 연결함
- Git 리포지토리에서 변경 사항을 5초마다 주기적으로 확인하고 변경사항이 감지되면 Airflow는 업데이트된 DAG 파일을 가져와 GKE 클러스터 내의 지정된 위치에 동기화가 진행됨
- 지정된 위치는 기본적으로 Airflow 스케줄러 및 워커 포드(pods)에 마운트된 PV(persistent volume)에 저장됨
- GitSync를 활성화하면 사이드카 패턴으로 동작함
- 생성된 사이드카 컨테이너는 코드 동기화를 처리하므로 기본 컨테이너는 애플리케이션의 핵심 기능 실행에만 집중 할 수 있어서 배포의 유지 및 관리를 단순화한다는 장점이 있음

(3) Airflow API 활성화

- DAG, 작업 상태, 실행 로그 등을 확인해볼 수 있는 환경을 구성함
- 환경변수와 커넥션 정보를 airflow web UI에 추가함으로서 민감정보가 깃허브에 노출되지 않게 작업함



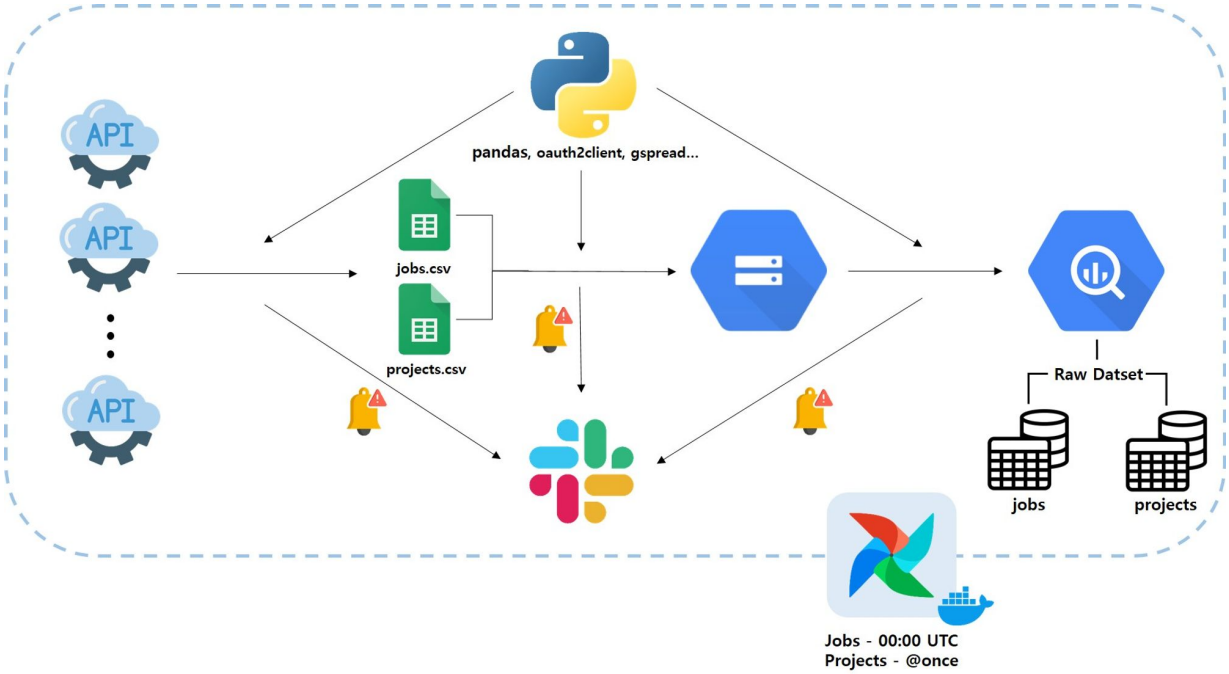
2) CI/CD

Git Action을 이용하여 dags 폴더안의 위치한 DAG의 코드를 검사하는 과정을 구현하였습니다. Git Action으로 다음 항목들을 검사합니다.

1) requirements.txt로 모듈 종류 및 버전 체크
2) flake8로 문법/스타일

B. Airflow를 이용한 ETL

API를 통해 데이터를 추출 전처리하여 구글 스프레드시트에 CSV로 저장 후 구글 클라우드 스토리지에 적재합니다. 이후 전처리하여 Bigquery에 적재합니다. 위 ETL 과정을 DAG로 작성하여 Airflow에 의해 정기적으로 실행됩니다.



1) 데이터 수집

[노인 일자리 공고 데이터]

공공데이터포털에서 2022년 6월부터 현재까지 한국노인인력개발원의
노인 구인정보 데이터를 수집하였습니다.

(1) OpenAPI (링크)

- '100세누리 구인정보 목록 검색'
- '100세누리 구인정보 상세 정보 조회'

(2) 수집 과정 :

a) DAG 실행 전일 기준으로 구인정보 목록을 API로 가져와
구글스프레드시트 내 'job_list_crawl 시트로 저장
b) 채용공고 ID를 사용하여 구인정보 상세 정보 API를 병렬로 호출
c) 상세 공고 정보를 받아와 'job_detail_crawl' 시트에 저장
d) 이후 데이터 전처리 수행 후 두 시트를 jobs 시트로 통합

[노인 일자리 사업 데이터]

공공데이터포털에서 2018년 부터 2023년까지 한국노인인력개발원의 노인 일자리
사업 통합 정보 데이터를 수집하였습니다.

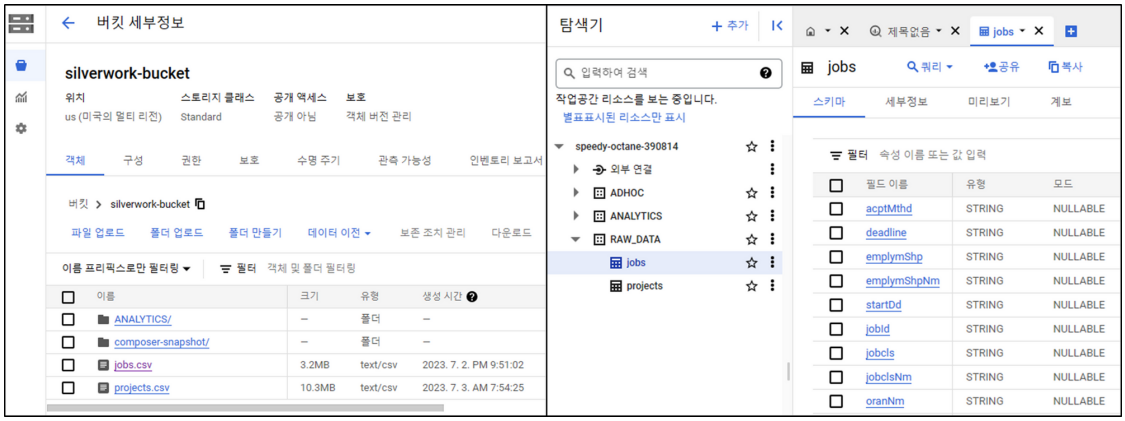
(1) OpenAPI (링크)

- '노인일자리사업 통합정보'

(2) 수집 과정 :

a) 3개년 일자리 사업 통합 정보를 API로 가져옴
b) 데이터 전처리 후 전체 데이터를 구글 스프레드시트 내 'projects'
시트에 저장

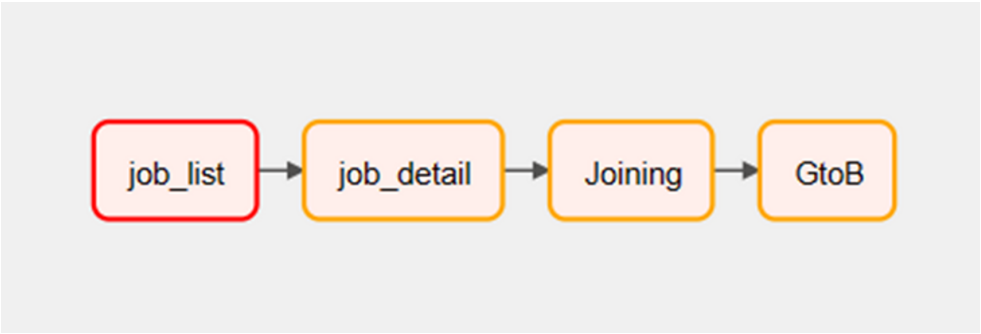
2) 데이터 적재



- Google Sheets 내 노인 일자리 공고와 사업 데이터를 추출하여 Google Cloud Storage → 버킷(silverwork-bucket)으로 silverwork 아래 각각 jobs.csv(공고), projects.csv로 업로드
- BigQuery에서 raw_data 데이터 셋(Data set) 아래에 jobs(공고), projects(사업) 테이블 생성
- Google Cloud Storage에 있는 jobs.csv, projects.csv를 각각 jobs와 projects 테이블에 벌크 업데이트 실행

3) Airflow DAG

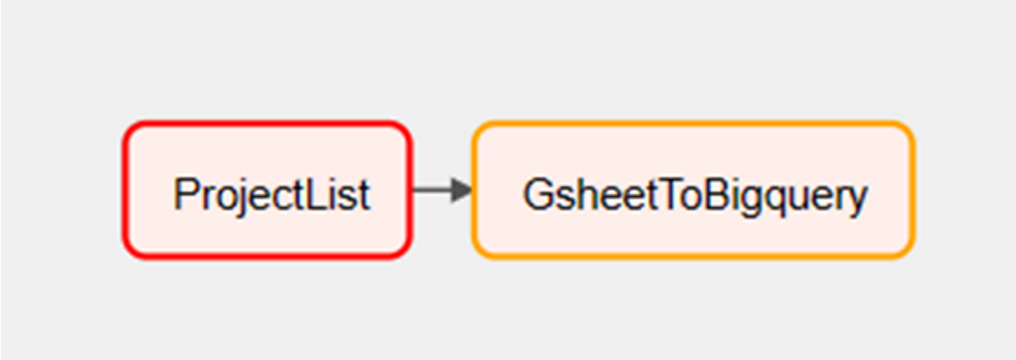
[jobs_crawling.py]



- 노인 일자리 공고 데이터 수집을 실행하는 DAG파일로 매일 자정(UTC 기준)에 한번씩 ETL 시행

- DAG 실행 전일을 기준일로 하여 노인 일자리 공고 데이터 수집 실행
- job_list: 구인정보 목록을 API로 가져와 구글스프레드시트 내 'job_list_crawl' 시트로 저장
- job_detail: 채용공고 ID를 사용하여 상세 공고 정보를 받아와 'job_detail_crawl' 시트에 저장
- Joining: 'job_list_crawl'와 'job_detail_crawl' 시트 내 데이터를 전처리 후 'jobs'시트로 통합
- GotB: 'jobs'시트 내 데이터를 구글 클라우드 스토리지에 'jobs.csv'로 업로드 후 BigQuery raw_data 데이터셋 밑에 jobs테이블로 벌크 업데이트 시행

[projects_crawling.py]



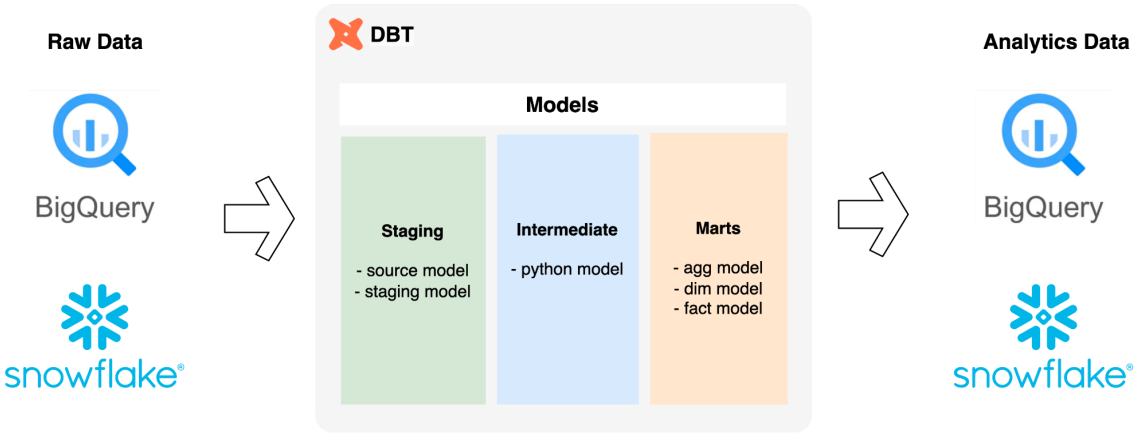
- 노인 일자리 사업 데이터 수집을 실행하는 DAG파일로 DAG 실행 시 한번 만 ETL 시행
- 노인 일자리 사업 데이터가 공공데이터 포털에 업로드 되는 날짜가 불명확함에 따라 노일 일자리 사업 데이터 업로드 확인 후 DAG 실행
- ProjectList: 사업 목록을 API로 가져와 구글스프레드시트 내 'projects' 시트로 저장
- GsheetToBigquery: 'projects'시트 내 데이터를 구글 클라우드 스토리지에 'projects.csv'로 업로드 후 BigQuery raw_data 데이터셋 밑에 projects테이블로 벌크 업데이트 시행

C. DBT를 이용한 ELT

Airflow 와 DBT, Docker 를 이용해 ELT 프로세스를 구축하였습니다

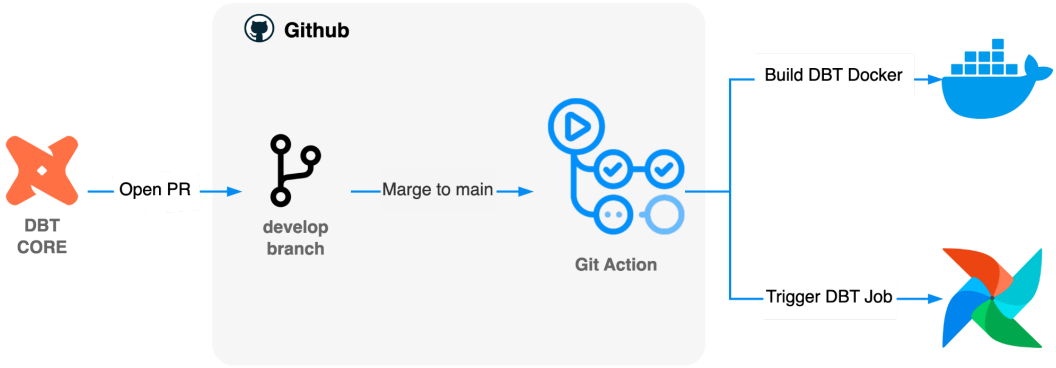
1) DBT ELT 프로세스 구축

- 모델링 레이어를 3단계로 나누어, 단계별로 가공과 품질을 검증합니다



1. 스테이징 모델
    - 원시 데이터를 가공하여 스테이징 모델을 정의
    - JOIN과 집계함수를 제외한 데이터 가공만 수행
2. 중간 모델
    - 스테이징 모델을 기반으로 집계함수, JOIN 및 복잡한 가공 처리
    - Google Cloud DataProc , Pandas를 이용한 Python 모델 처리
3. 마트 모델
    - 중간 모델과 스테이징 모델을 활용하여 분석팀 요청에 맞는 모델 정의
    - fact 모델은 Incremental update를 수행

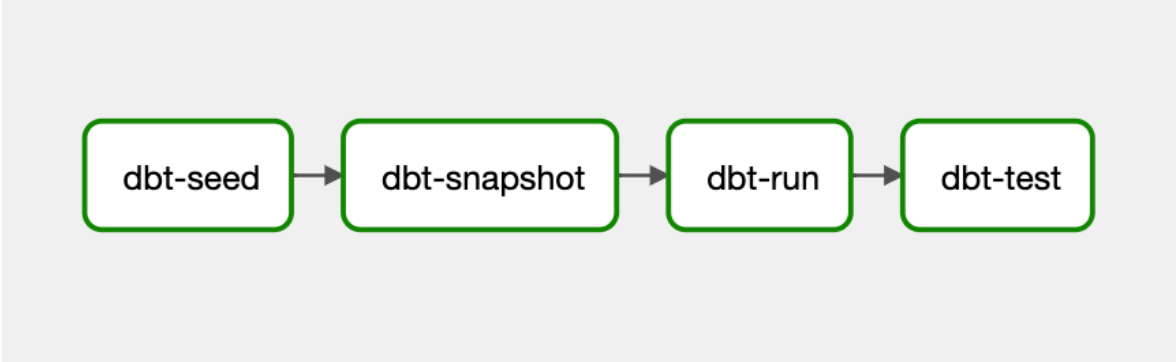
2) DBT 배포 프로세스 구축



- Airflow 와 코드베이스를 분리하기 위해 DBT 프로젝트를 별로도 생성
- develop, main PR 이벤트가 발생시, dev, stage DW에 모든 모델을 생성하고 검증
- main 브랜치 push 이벤트가 발생시, 도커이미지 생성 및 Airflow DAG 트리거

3) DBT DAG 생성

- Airflow KubernetesPodOperator를 이용해 DBT 이미지를 가져와 DBT 명령 수행
- 매일 1회 DAG를 수행
- 에러가 발생할 경우 Slack 채널에 알림

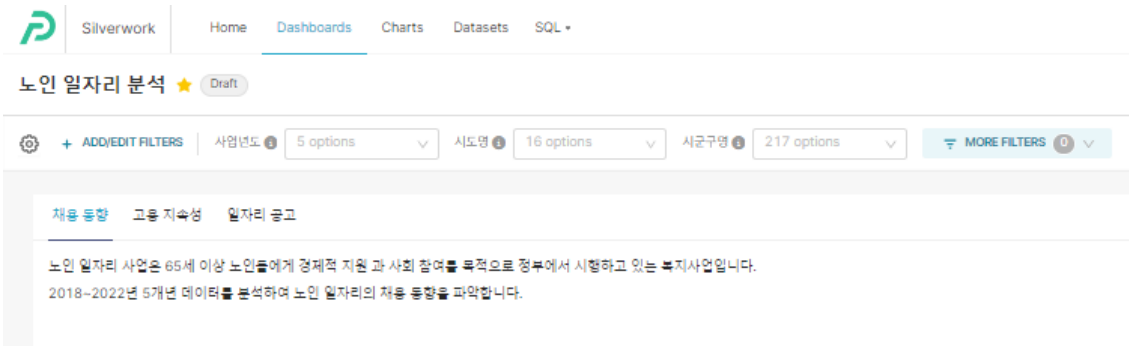


D. 대시보드 개선

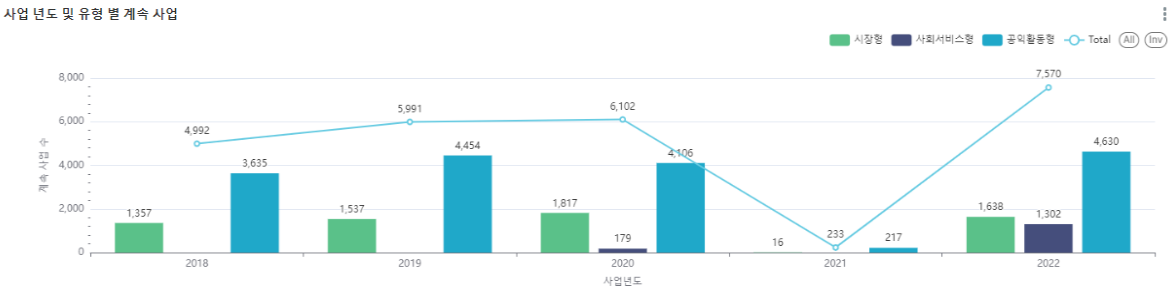
기존 대시보드의 낮은 차트간 연관성 및 커스텀이 불가한 차트를 개선했습니다

1) 대시 보드 가시성 향상 탭

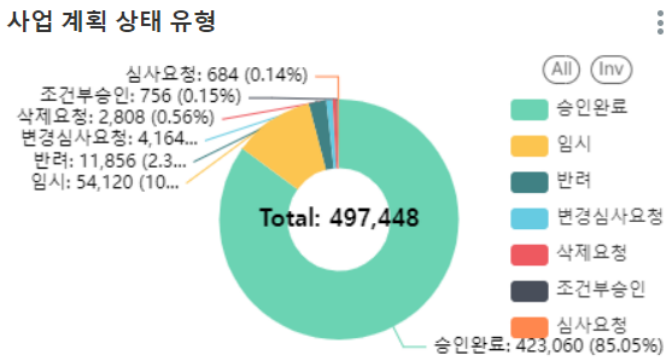
(1) 대시보드 탭 분리, 차트 간의 연관 관계를 쉽게 파악할 수 있다



(2) 지표 정보 추가

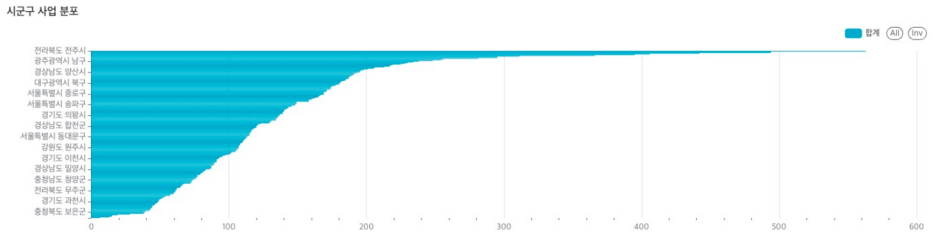


[그림1-1] x축 y축 타이틀 및 legend 목록을 추가한 막대 차트

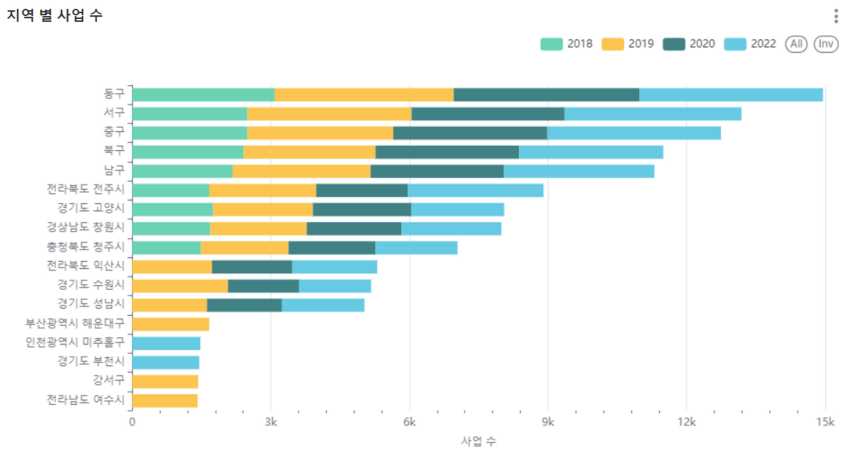


[그림 1-2] legend 및 Total Value를 추가한 파이 차트

(3) 그룹 정보 추가

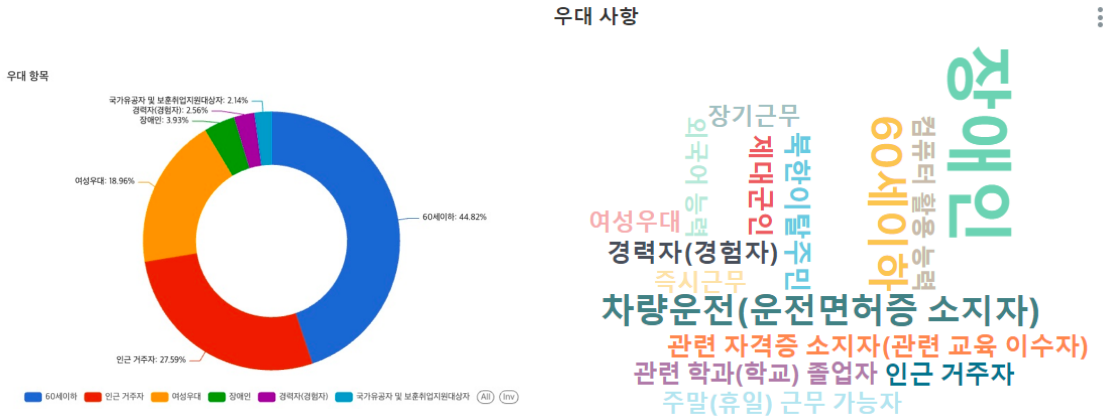


[그림 1-3]그룹 적용 전



[그림 1-4] 사업 년도로 그룹화, 지역 및 사업 년도별 사업 수 분석

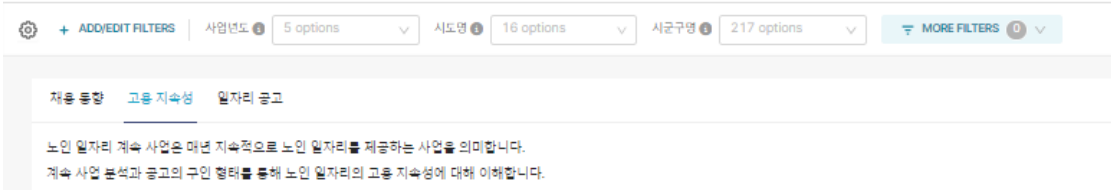
(4) 차트 유형 변경



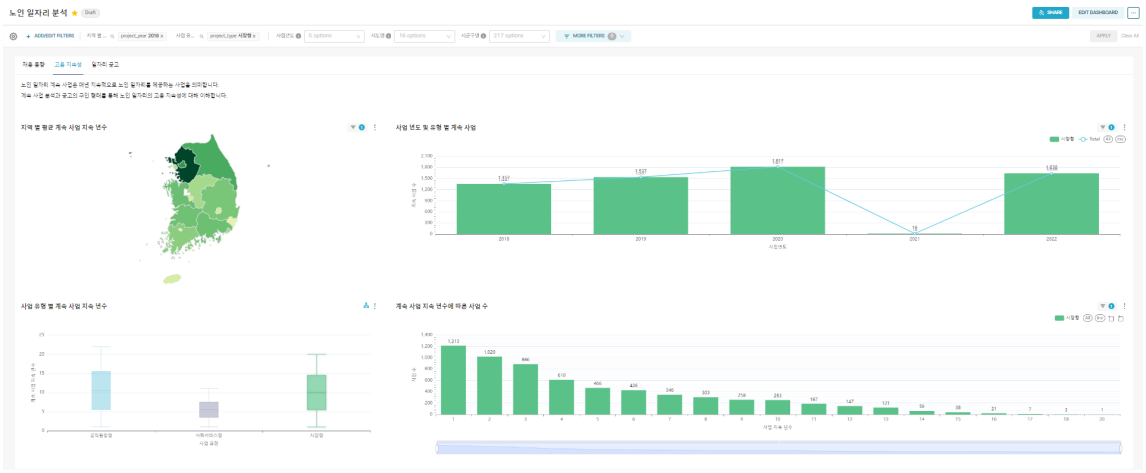
[그림 1-5] 우대 사항 값이 명확한 WordCloud 차트로 변경

2) Interactive Chart 구현

(1) 차트 필터 및 차트 연동(Cross-Filtering)

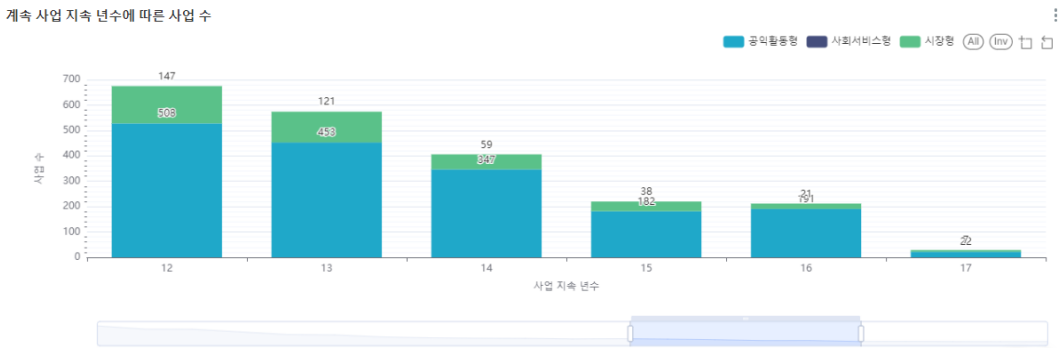


[그림 2-1] 대시 보드 상단의 필터



[그림 2-2] 지도내 지역을 클릭하면 연결된 차트의 결과도 변경

(2) 데이터 확대 (Zoom 옵션)



[그림 2-3] Zoom기능이 추가된 '계속 사업 지속 년수에 따른 사람 수' 차트

6. 참여자 정보 및 각 역할

이름	담당업무
김상희	대시보드 고도화
김혜민	GCP 인프라 구축
남윤아	Airflow를 이용한 ETL 구축
이성희	DBT를 이용한 ELT 구축
이하윤	GCP 인프라 구축

7. 프로젝트 결론 및 회고

1) 아쉬운 점 및 개선 사항

[GCP 인프라 구축]

- GKE 클러스터 비공개 클러스터로 변경
- Airflow 메타 데이터베이스를 Cloud SQL로 변경
- Google Secret Manager 사용
- GKE 환경 설정 추가

[DBT를 이용한 ELT구축]

- DBT 메크로와 데이터 카탈로그를 이용한 DAG(Model) 추상화

2) 느낀점

수집된 데이터에서 유의미한 차트를 구성하는 것이 어려웠습니다. 데이터가 어떤 것을 의미하는지를 파악하는 것이 중요하다고 느꼈습니다.

GCP와 인프라 구축은 처음이라 이해하는 데에 시간이 좀 걸렸습니다.
그 과정에서 많이 환경 리셋을 했지만, 원하는 구현을 해내기도 했습니다.
공부를 많이 해가며 완성한 프로젝트였던만큼 얻어가는게 많은 프로젝트였습니다.

DBT를 공부하며 수집된 데이터를 어떻게 처리해야 하는지,
데이터 품질에 대해 고민할 수 있던 프로젝트였습니다
쿠버네티스 환경에서 개발해 분산환경에 대해 이해할 수 있었습니다

k8s에서 Airflow를 배포하는 법과 인프라 구성,
GCP 서비스에 대해 학습할 수 있어서 의미가 있었습니다.
러닝커브가 있어 본래 계획대로 구축하지 못했지만,
이는 추후 개선해볼 생각입니다.