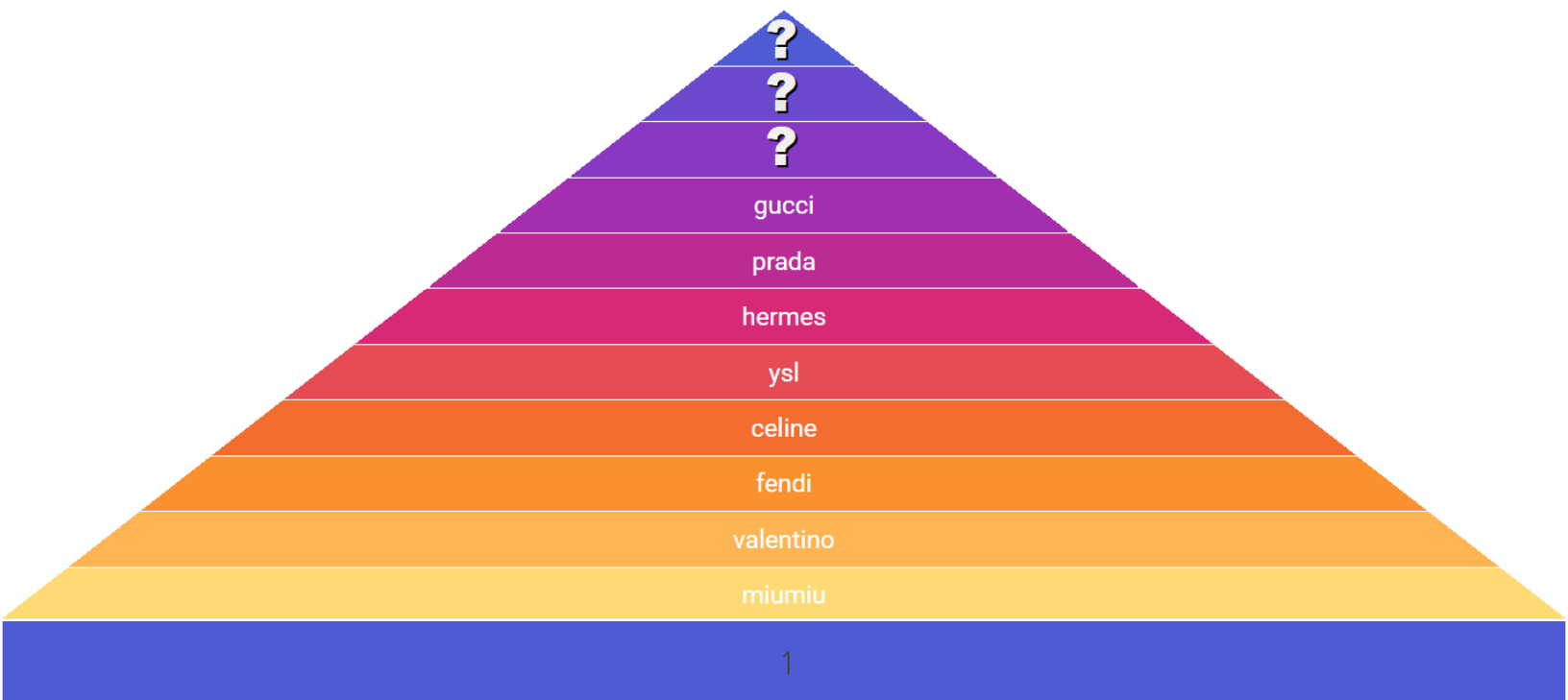


2팀 1조

명품 브랜드 인스타그램 화제성 분석

이성희, 이하윤, 임형우



목차

1. 프로젝트 개요
2. 프로젝트 주제 선정 이유
3. 참여자 정보 및 역할
4. 활용 기술 및 프레임워크
5. 프로젝트 세부 결과
6. 프로젝트 결론 및 회고

1. 프로젝트 개요

명품 브랜드의 인스타그램 계정과 명품 브랜드 해시태그 검색 결과를 수집하고, 이를 분석하여 명품 브랜드의 화제성을 평가합니다

2. 프로젝트 주제 선정 이유

명품 브랜드의 인스타그램 계정과 관련 해시태그를 수집하여 브랜드의 활동성과 인지도를 측정하고, 각 브랜드의 고유한 온라인 마케팅 전략을 파악합니다. 이를 통해 브랜드 간의 차별화를 확인하고 경쟁 우위를 평가하며, 소비자 동향을 파악하는 중요한 정보를 얻을 수 있습니다.

3. 참여자 정보 및 역할

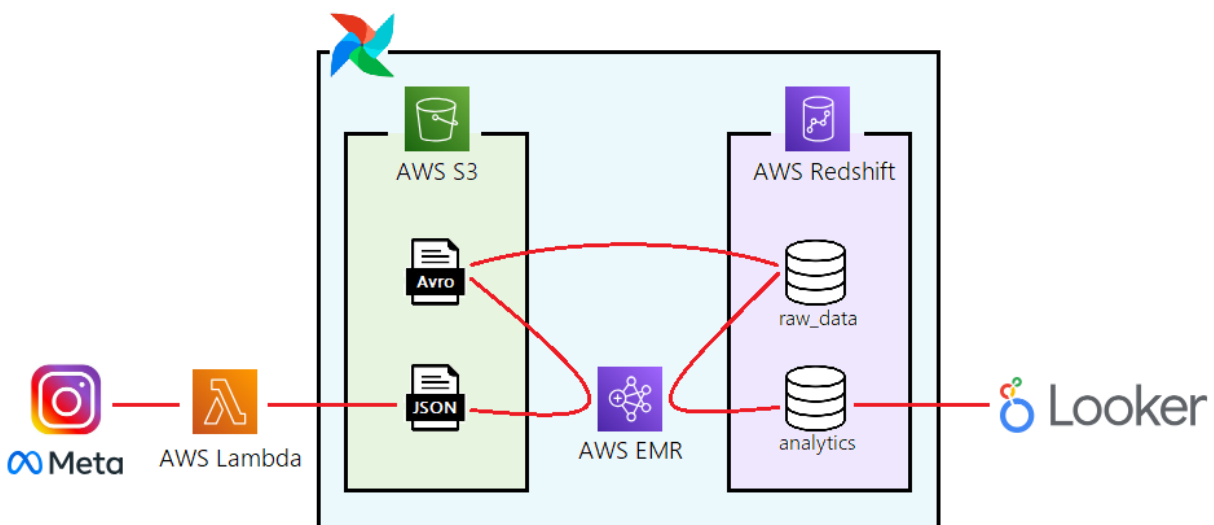
업무	이성희	이하윤	임형우
기획	데이터 모델링	- 데이터 모델링 - 분석지표 정의	데이터 모델링
Infra	- AWS 네트워크 관리 - 데이터/컨테이너 환경 구축 - Airflow CI/CD	-	- Lambda 환경 관리 - 스크래퍼 CI/CD
Scrapping	-	-	- Instagram API 스크래퍼 - Lambda Event 스케줄링
ETL	- Airflow ETL 스케줄링 - EMR 프로세스 개발 - Slack 모듈 개발	-	-
ELT	-	- 데이터 마트 모델링 - 마트 쿼리 작성 - Airflow ELT 스케줄링 - EMR 프로세스 개발	- 데이터 마트 모델링 - 마트 쿼리 작성
Visualization	-	Looker 대시보드	Looker 대시보드

4. 활용 기술 및 프레임워크

A. 개발 환경

Field	Stack
Infra	AWS Secrets Manager AWS Cloudwatch AWS EC2
Scrapping	AWS Lambda
ETL & ELT	Airflow Spark pydeequ
Data Storage	AWS S3 AWS Redshift Snowflake Amazon RDS
BI tool	Looker
CI/CD	Docker Github Actions AWS ECR AWS ECS
ETC	Slack Notion

B. 프로젝트 아키텍처



C. 수집계획

1) 수집 기간 : 2023-08-28 ~ 2023-09-02 (6일)

2) 수집 전략

11개 명품 브랜드에 대해서 각 API 요청.

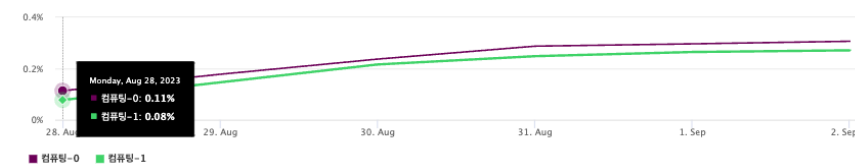
- 인스타 계정/미디어 1시간 당 API 요청 1회
 - 요청 당 계정 데이터 1레코드, 미디어 데이터 25레코드 발생
- 해시태그 검색 5분 당 API 요청 1회
 - 요청 당 해시태그 미디어 데이터 25레코드 발생

3) API 호출 현황

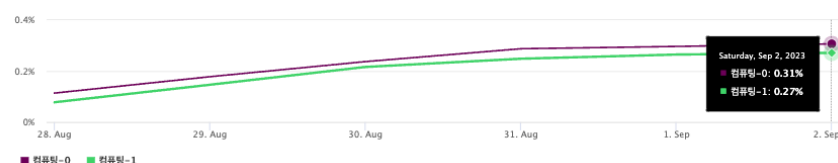
- Total Objects: 46944건 (1일 7824)
- Total Size: 375 MB (1일 62.5 MB)

4) Redshift 디스크 공간 비율 변화 (총 320 GB)

사용된 디스크 공간 비율(%)
사용된 디스크 공간의 비율(%)입니다.

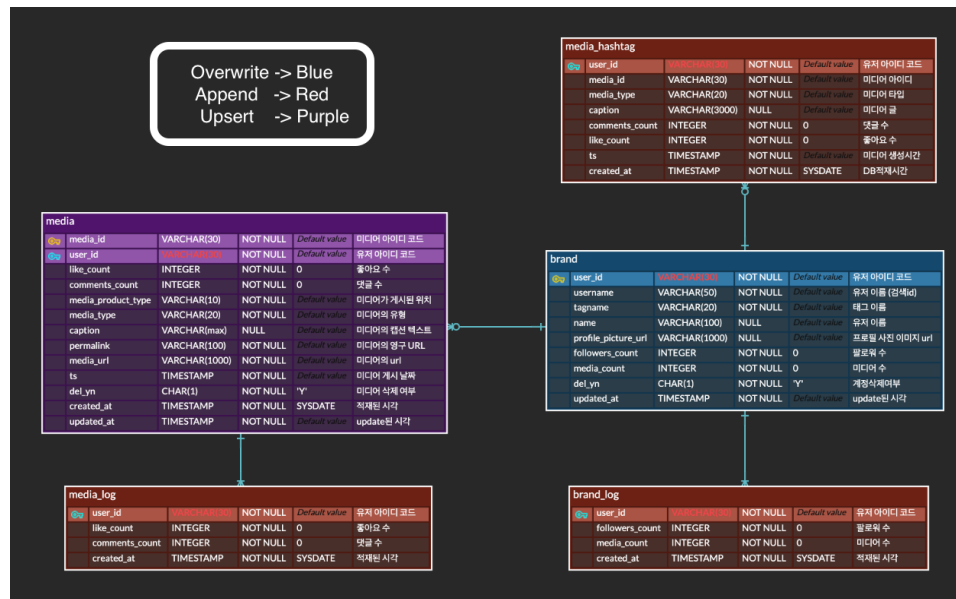


사용된 디스크 공간 비율(%)
사용된 디스크 공간의 비율(%)입니다.

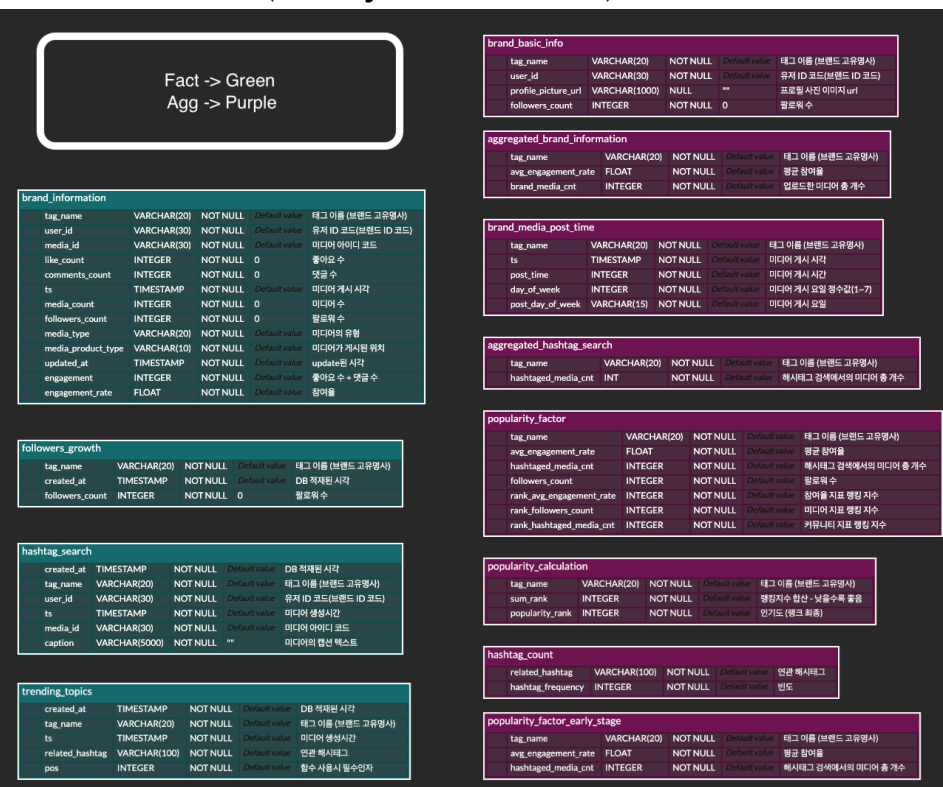


D. Schema

< RAW_DATA Schema >



< Analytics Schema >

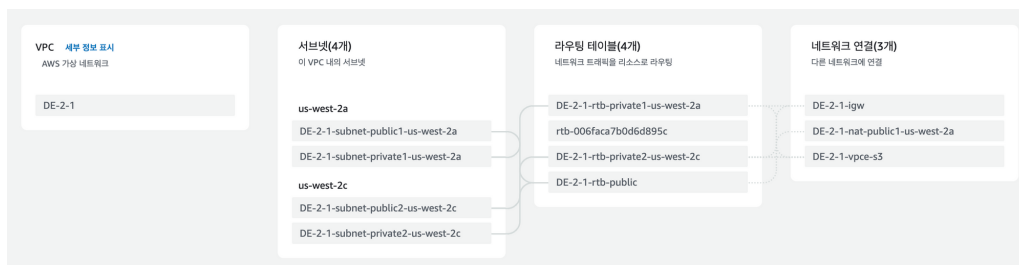


5. 프로젝트 세부 결과

A. AWS 인프라

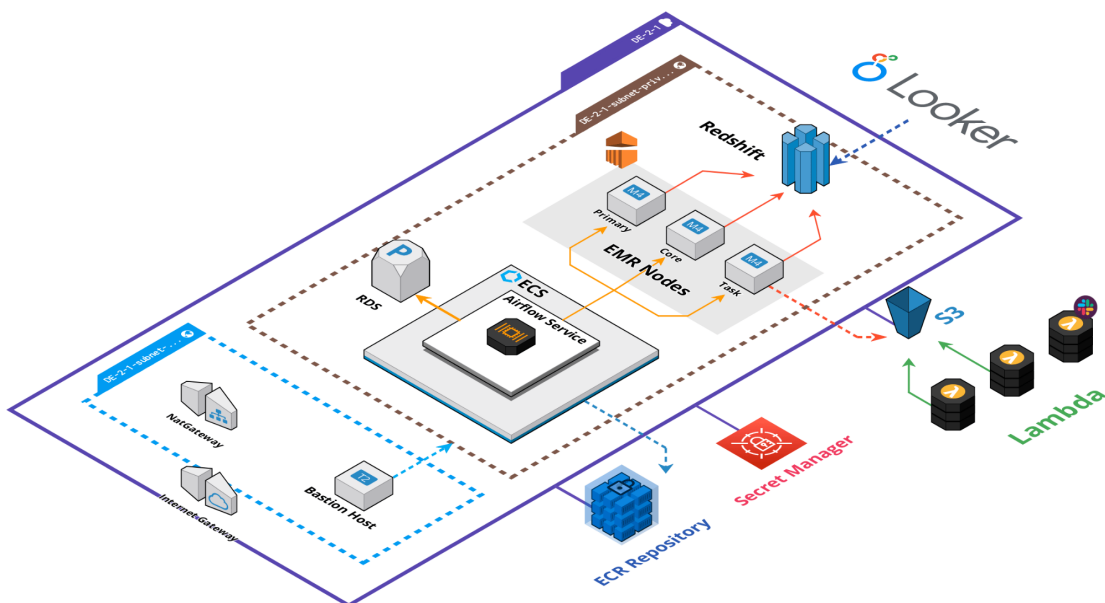
- 기본적인 네트워크 인프라 구축
- S3 , Redshift, EMR 클러스터를 구축하여 대규모 데이터 처리 환경 제공
- ECR , ECS를 사용하여 Airflow 배포 프로세스를 구축

1) 네트워크 인프라



- 리전별 가격 비용이 저렴한 us-west-2 을 선정하여 VPC를 구축
- 2개의 가용영역에 각각 Private Subnet , Public Subnet 생성
- Private Subnet의 서비스에 접근할 수 있도록, Bastion Host 인스턴스를 생성

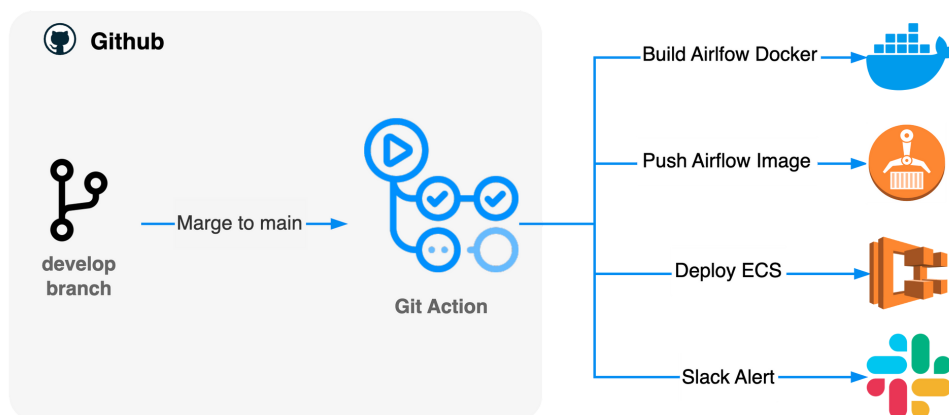
2) 데이터 파이프라인 서비스



S3	환경변수 및 EMR 저장소, 데이터 레이크
Lambda	데이터 수집 프로세스 실행 및 스케줄링
Secret Manger	토큰, Connection 등 민감 정보 관리
EMR	ETL, ELT 작업을 처리하는 클러스터 - m4.large Spot, 최소 노드 2 (Primary, Core)
Redshift	Raw Data , Mart Data가 적재된 데이터 웨어하우스 - dc2.large, 2노드
ECR	Docker 이미지 저장소
ECS	Airflow 컨테이너를 관리하는 Serverless 오케스트레이션 - fargate Spot, 4개 컨테이너
RDS	Airflow 서비스의 메타 데이터 저장소

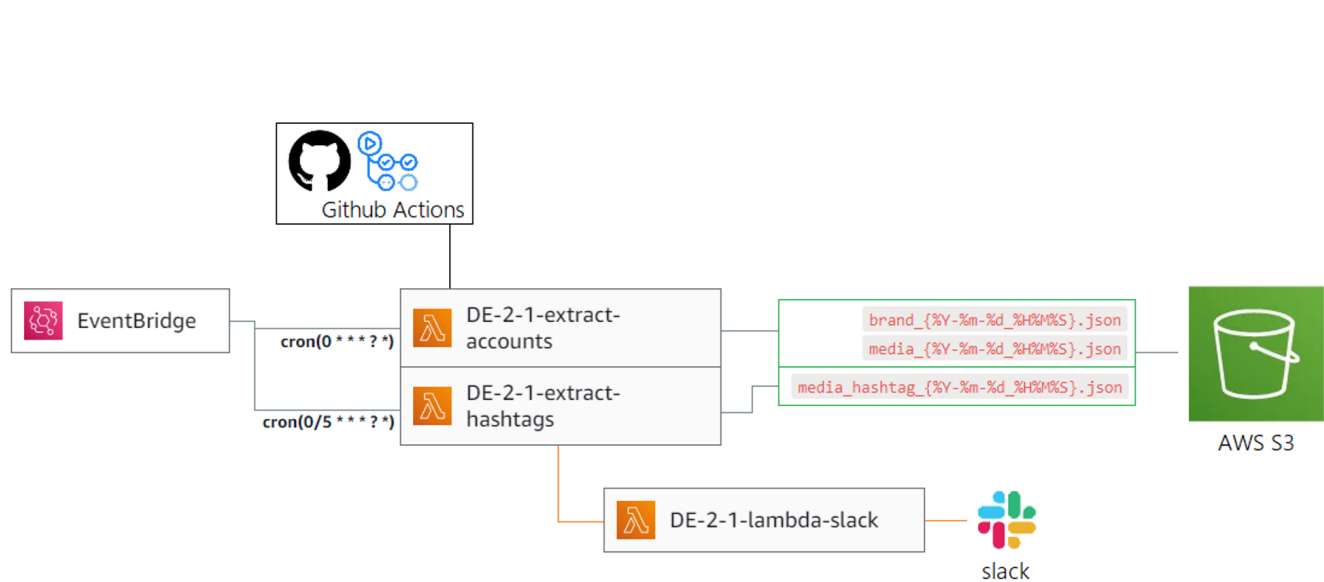
3) Airflow 서비스 배포 프로세스

- Github Action을 이용하여 Docker 이미지를 생성해 ECR에 저장
- Airflow 빌드 정보가 담긴 task-definition의 이미지 버전정보 업데이트
- ECS를 통해 Airflow 서비스 배포 (롤링 업데이트)
- 컨테이너별 Health Check 후 정상이 아닐 경우 재배포
- 배포 결과 Slack으로 알람



B. Lambda 수집기

- Instagram Graph API를 통해 데이터 수집기 개발
- API 응답 데이터 검증 및 필드 전처리 후 JSON 포맷으로 S3 적재
- Github Actions를 이용하여 CI/CD 및 AWS Lambda 배포




1) 데이터 수집 및 저장

- 1시간 마다 프로필 API를 통해 프로필 데이터와 최신 25개의 미디어 데이터 수집
- 5분 마다 해시태그 검색 API를 통해 최신 25개의 해시태그 미디어 데이터 수집
- brand, media, media_hashtag 세가지 모델로 데이터를 저장
- 응답 필드가 존재하지 않는 경우 default값을 부여

2) 데이터 품질 점검 및 로깅

- 수집대상, 맵핑 필드, 저장 포맷 등 메타데이터 configure.ini 파일로 일괄 관리
- 변경 또는 확장할 경우를 가정하여, 설정파일을분리하여 관리
- 필수 데이터필드가 존재하지 않거나 예상하지 응답값은 적재 제외 처리
- 수집 된 각 데이터에 대한 품질사항을 점검하기 용이하도록 로깅 설계
 - a. 태스크 실행 시간
 - b. 생성된 수집 데이터 파일위치
 - c. 레코드 또는 필드에 대한 데이터 이상 내역 출력

🕒 2023-09-03T04:00:00Z job Success 
 resource: ['arn:aws:events:us-west-2:862327261051:rule/DE-2-1-extract-accounts-job']
 FILE[de-2-1-s3/raw/brand_2023-09-03/brand_2023-09-03_040015.json, de-2-1-s3/raw/media_2023-09-03/media_2023-09-03_040015.json]

===== ERROR =====


luxury_media-[not found field] media_url(media_url) 2

luxury_profile-[not found field] name(name) 1

luxury_media-[not found field] caption(caption) 2

=====

```
{
  "statusCode": 200,
  "start_time": "230903_0400_15",
  "end_time": "230903_0400_50",
  "body": "\"end processing\\nupload success\""
}
```

🕒 2023-09-03T04:00:00Z job Success 
 resource: ['arn:aws:events:us-west-2:862327261051:rule/DE-2-1-extract-hashtag-job']
 FILE[de-2-1-s3/raw/media_hashtag_2023-09-03/media_hashtag_2023-09-03_040047.json]

===== ERROR =====

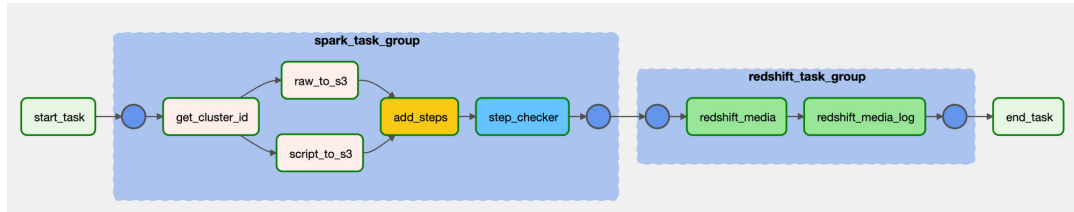
hashtag_media-[not found field] like_count(like_count) 12

=====

C. ETL Process

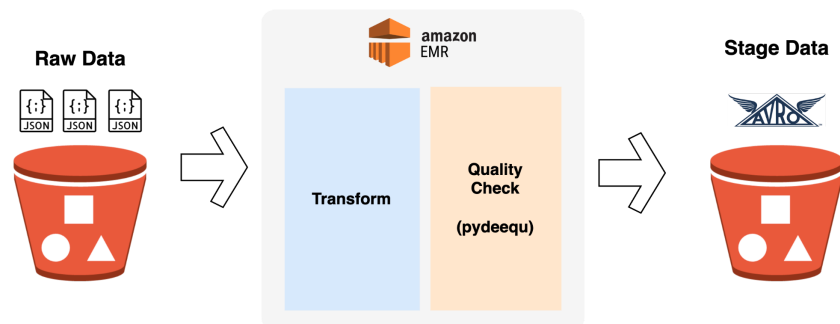
- Ariflow를 통해 ETL 작업 스케줄링
- AWS EMR을 이용한 Raw Data 가공, 검증 및 Redshift 적재

1) Airflow 스케줄링



- 매 시 정각마다 Raw Data를 Stage Data로 처리하는 DAG
- 동적 DAG 를 구현하여, 수집 데이터 모델 별로 DAG 생성
- Slack 알림 모듈을 구현 및 각 task 의 결과를 Slack 으로 전송
- EMR을 이용하여 데이터 처리 및 품질 체크 후 Stage 데이터 생성
- 모델별 저장 전략에 따라 Stage 데이터를 Redshift Raw Schema의 테이블로 적재

2) EMR 프로세스



- Raw Data와 EMR 클러스터가 수행할 스크립트 정보를 S3에 적재
- **pydeequ** 라이브러리를 이용해 데이터 품질체크 수행
- 처리 결과 S3 /Stage 버킷에 Avro 타입으로 저장 (Avro : 자동매핑 편의성 제공)
- Sensor를 통해 EMR 작업의 결과 확인

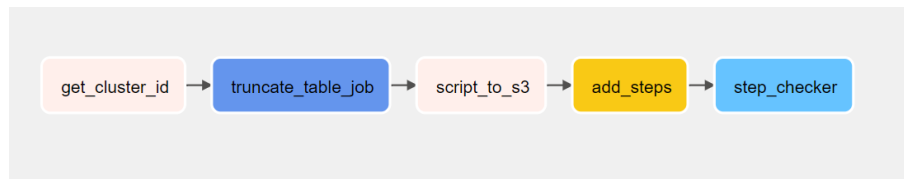
D. ELT Process

- Ariflow를 통해 ELT 작업 스케줄링
- EMR을 통해 Redshift의 Raw Data 검증 및 Mart Data로 가공 및 적재

1) ELT 모델링

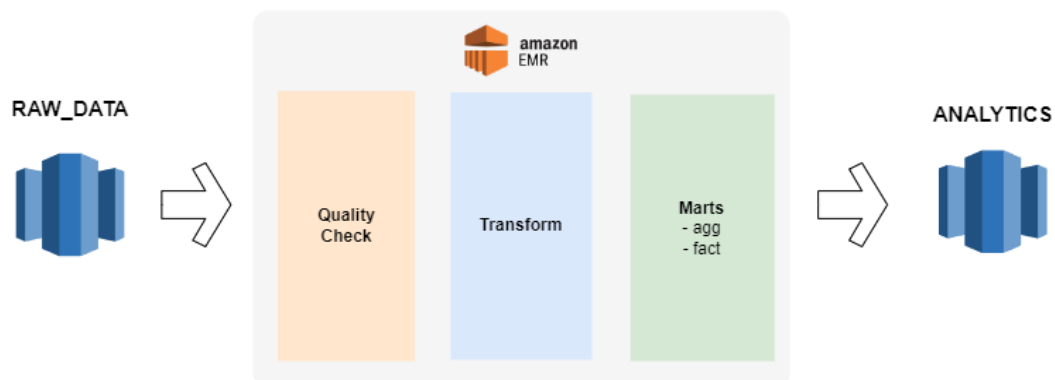
- 분석 목표를 바탕으로 대시보드의 차트에 쓰일 데이터 마트 모델링
- EMR 프로세스를 통해 데이터 마트 생성

2) Airflow 스케줄링



- 매 시 30분 마다 Raw Data를 Mart Data로 처리하는 DAG
- EMR을 통해 데이터 가공, 품질 검증, 데이터 적재 작업 수행
- Slack 모듈을 활용하여 각 task 의 결과를 Slack 으로 전송

3) EMR 프로세스



- 품질체크 쿼리를 통해 Raw 데이터를 검증 후 로딩 및 실패시 Dag 종료
- 마트 테이블 모두 Overwrite 적재 수행
- Cross Filtering 차트를 위한 Fact 테이블에 데이터 적재
- 데이터 집계 연산 후 집계(Agg) 테이블에 적재
- 분석 지표에 맞춰 데이터 가공 후 Fact 테이블에 각각 적재
 - a. 인스타 미디어 Caption 데이터에서 해시태그 키워드 추출
 - b. Timestamp 데이터를 시간, 요일 데이터로 각각 변환
 - c. 인기도 지표에 필요한 3가지 지수를 계산 후 랭킹 처리

E. Dashboard

- 분석 지표를 정의 및 대시보드 기획
- 종합 차트 및 반응형 차트를 구현을 통한 가시성 확보

1) 분석 지표 정의

- 참여율(%) = (댓글 수 + 좋아요 수) / (팔로워 수) * 100
- 인기도 = (미디어 지수 + 팔로워 지수 + 커뮤니티 지수) 지표 합산 랭킹
 - a. 인기도 지표를 정의해서 명품브랜드 11개의 실시간 변동 순위 제시
 - b. 미디어 지수 : 브랜드 인스타그램 미디어의 평균 참여율에 따라 랭킹 부여
 - c. 팔로워 지수 : 브랜드 인스타그램 팔로워수에 따라 랭킹 부여
 - d. 커뮤니티 지수 : 브랜드 이름이 해시태그된 미디어의 개수에 따라 랭킹 부여

c. 생성한 차트

■ Summary 차트

- 각 브랜드 별 집계한 정보들을 요약 제시
- 각 컬럼별로 높은 지표들은 색상 그라데이션을 주어 한 눈에 보이게끔 함
- 브랜드 간의 차별화를 확인하고 경쟁 우위를 평가

■ Most Frequent Post Types, Most Engaging Post Types, Most Frequency Video Post Types 차트

- 미디어 타입 비율 분포, 미디어 타입 별 참여율, 비디오 타입에서의 릴스랑 비디오 비율 분포를 알 수 있음
- 실제로 참여율이 높은 미디어를 피드에 많이 반영을 하는지, 아니면 그렇지 않은지, 릴스 같은 숏폼 콘텐츠에도 적극적으로 뛰어드는지 등의 팩트 체크를 함으로서 각 브랜드의 고유한 온라인 마케팅 전략을 파악 할 수 있음

■ Post Engagement Average Rate 차트

- 브랜드가 올린 미디어의 하루기준 평균 참여율을 나타냄
- 참여율에 비례하여 그래프 점들이 크게 보이게 함
- 소비자 동향을 파악

■ Brand Post Upload Activity 차트

- 브랜드가 미디어를 포스트하는 개수를 나타냄으로서 게시글 업로드 추이를 볼 수 있음
- 브랜드의 인스타그램 내에서의 활동성을 확인 할 수 있음

■ Instagram Post Upload Time Distribution 차트

- 브랜드가 미디어를 포스트하는 요일과 시간 분포를 나타냄
- 브랜드 총계에서는 금요일 12시가 가장 활발한 업로드 시간으로 나타나짐
- 요일 상관없이 대체적으로 12시와 오후 4시 경에 많은 업로드를 함

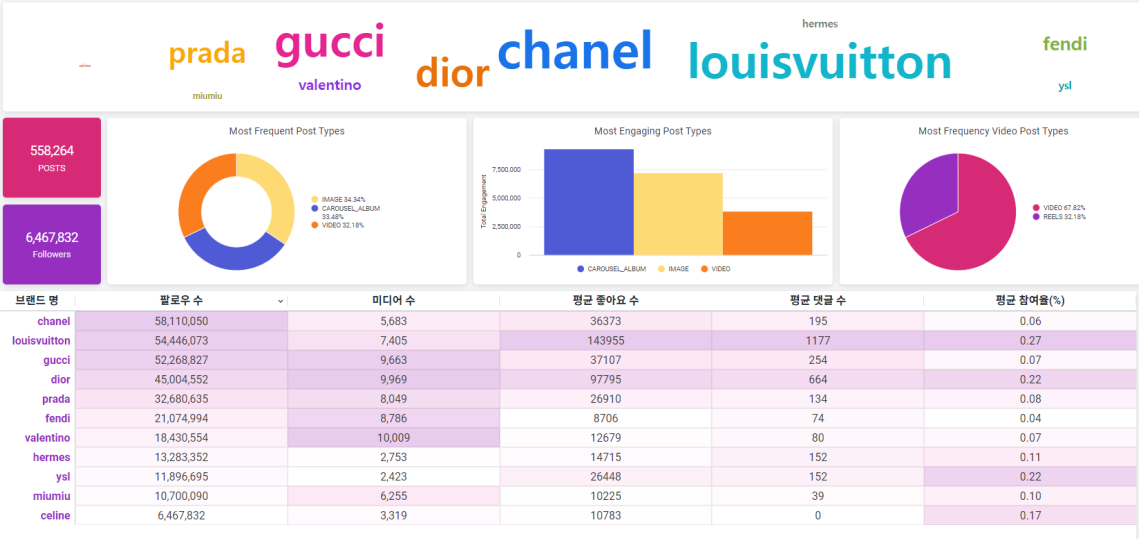
Detailed Analysis by Luxury Brands

미디어 업로드 날짜: 미디어 업로드 시간대: 미디어 업로드 요일

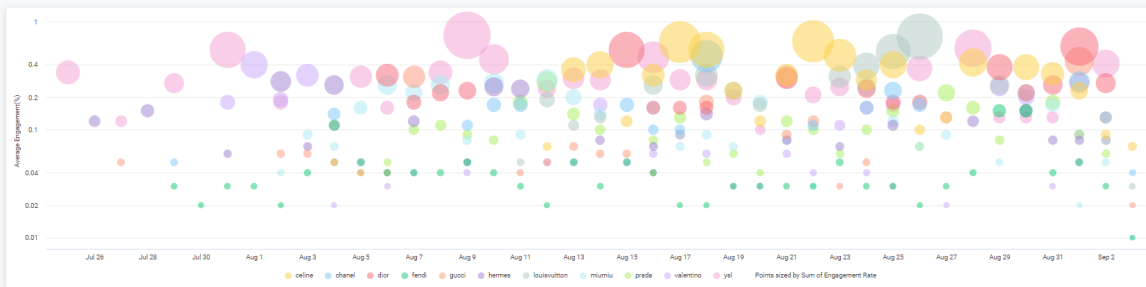
Last 30 Days 0 24 Is Monday or Friday or Tuesday or Wednesday...

Luxury Brand Analysis Project on Instagram

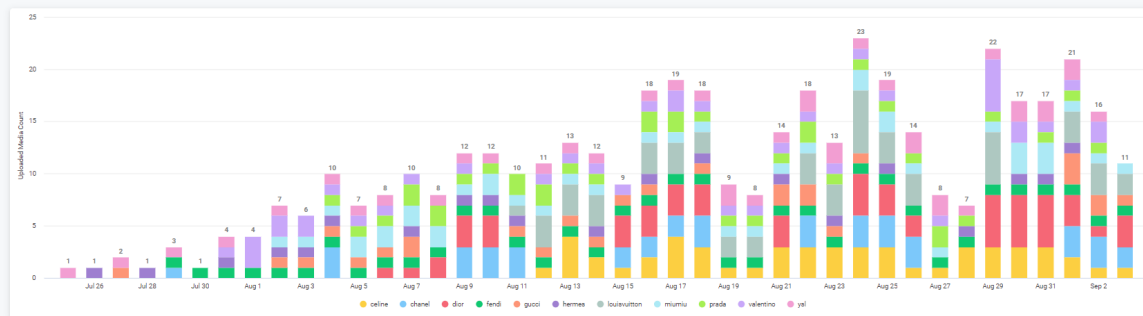
Instagram Graph API를 통해 명품 브랜드의 상세 분석



Post Engagement Average Rate



Brand Post Upload Activity



Instagram Post Upload Time Distribution

시간	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
요일	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Sunday	0	1	2	4	2	7	3	11	3	2	0	14	0	0	0	0	0
Monday	0	0	2	3	3	5	1	10	4	0	4	10	3	2	1	0	0
Tuesday	1	1	1	12	3	5	4	13	4	3	3	16	0	0	1	1	0
Wednesday	0	0	1	10	2	7	0	13	7	3	7	15	2	2	0	1	1
Thursday	0	0	5	10	6	5	3	17	4	1	7	14	2	2	0	2	2
Friday	0	0	4	7	8	4	4	23	6	5	2	16	3	0	0	1	0
Saturday	0	1	1	3	2	6	3	11	9	1	6	13	2	0	1	2	0

6. 프로젝트 결론 및 회고

1) 아쉬운 점 및 개선 사항

- 데이터 수집 프로세스에서 정상 수집과 에러 발생등의 로그를 보내는 채널이 똑같아서 특히 5분마다 스케줄되는 작업의 경우 문제가 발생하더라도 문제를 파악하기가 매우 힘들었다. 실패에 대한 채널을 분리하는 방법이나, 정상 작업의 경우 지속적 알림이 아닌 일정 시간마다의 리포트 형식으로 알림을 보내도 괜찮을 듯 하다.
- pydeequ 문서가 부족하고 버그가 있어 아쉬움이 있었다
SQL로 데이터의 품질을 검증하는 방향으로 개선할 수 있을것 같다
- 인프라 자원에 대해 모니터링 대시보드와 알람 구성을 구축하지 못했다
비용 단계를 두고 대응 전략을 세워야 할 것 같다
- 장기적으로 서비스를 운영한다면,
인프라 자원을 자동으로 on/off 하는 기능을 추가해야한다
현재 프로젝트처럼 배치간격이 있는 경우 task가 수행할 때만 클러스터를 생성하고 종료시키는 방법으로 발전할 수 있다
- 서비스를 운영하면서 축적되는 로그 데이터를 관리하는 방법이 필요하다
일정 주기를 정해 압축하여 s3에 백업하고 정리하는 방법이 있을 것 같다
- 수집, etl, elt의 상위에서 데이터의 메타 정보를 관리하는 데이터 디렉터리가
구축되었다면 개별 관리해 싱크를 맞추는 불편함을 줄일 수 있을 것이다
- S3 버के닝 전략을 반영하여(Y/M/D) 파티셔닝 성능을 높일 수 있다
- Airflow는 스케줄러로, 직접 수행하는 PythonOperator 및 기존 Operator를
대체해 Airflow의 부하를 줄여야 한다. 현재 프로젝트에서는 무리가 없었지만, 더 큰
규모의 데이터를 처리한다면 DockerOperator , Kubernetes Pod Operator를
사용하면 작업을 격리할 수 있다
- ELT 과정의 품질체크에서, EMR stdout에 결과를 출력하고 실패 시 예외처리를 하여
마무리 했지만, 품질체크 결과를 별도의 테이블에 저장하여 이후 SQL을 이용하여
품질체크 이력들을 조회하고, 2차적으로 분석하여 서비스 품질을 높일 수있도록
발전시킬 수 있다.

2) 느낀점

- Lambda를 사용함에 있어서 단순히 코드를 업로드만 하면 끝날 줄 알았으나, 패키지 사용, Lambda 캐싱 이슈 등의 문제가 발생하였다. 캐싱이슈의 경우 다른 방법을 이용하여 배포문제를 해결하긴 했으나, 근본적인 해결을 하고 싶었으나 시간 부족으로 하지 못해서 아쉽다. 재현도 해 보았으나 발생하지 않았고, 누구도 이유를 알 지 못했다.
- EMR에서 Redshift로의 테이블 안의 데이터를 overwrite하는 작업에서 시간과 부하가 많이 걸렸던 이슈가 있었는데, 이 과정에서 Redshift, EMR, Spark, Cloudwatch의 메트릭 추적을 통해 논리적으로 분석해야함을 알았다. 이 부분에 대해서 잘 학습하여 다음에 동일한 이슈가 생긴다면 논리적으로 해결해보고 싶다.
- 데이터 마트 모델링 하는 과정에서, 분석에 직접적으로 쓰이지 않는 테이블을 전부 Redshift에 overwrite하는 것은 지양해야하는 일이라는 것을 알았다. Spark df를 적극적으로 활용하여, 차트가 최종적으로 바라보는 데이터만 Redshift에 적재하는 것으로 후반에 바꿨고, 이 부분에 대해 신경써야한다는 것을 깨달았다.
- ELT에서 스파크를 통해 데이터를 로드, 가공, 적재하는 과정에서 Spark df를 이용해서 대부분 처리했다. Spark sql을 통해서 처리해보는 학습시간을 가지면 좋을 것 같다.
- Spark를 python으로 사용함에 있어 레퍼런스의 경우 Scala에 비해서 상세한 설명이 부족하다고 느꼈다.
문제상황에 직면해서 해결책을 찾는데에도 커뮤니티에서 비슷한 상황을 찾는 것도 쉽지않아 불확실한 것도 직접 시도 해보며 개발해야해서 매우 어려움을 느꼈다.
- 데이터 모델링과 데이터 가공 및 적재에 신경을 쓰느라 연관된 인프라 자원에 대해 신경을 많이 못 쓴 것 같다. AWS 웹에서 Cloudwatch를 통해 인프라 자원에 대해 모니터링 했으나, 추가적으로 알람 시스템을 구축했으면 더 좋았을 것 같다.