1. **프로젝트**

* **주제**: 토트넘 경기 결과 분석 및 데이터 시각화
* **목적:** Python과 SQL을 활용하여 데이터를 수집하고, 분석 후 웹 대시보드로 시각화
* **사용 기술 스택:**
  + Python (pandas, Flask)
  + MySQL
  + HTML, CSS, JavaScript (Chart.js, jQuery)

1. **구현 과정**

**2-1. 데이터 수집**

* A.HLL-SRC-2315726-인공지능공학부-이가은.ipynb 파일에서 Football Data API를 사용하여 2022-2024 시즌 토트넘 경기 데이터를 수집.
* API 호출을 통해 경기 날짜, 상대 팀, 장소, 득점 결과 등의 데이터를 가져와 CSV 파일로 저장한 후 MySQL 데이터베이스에 저장
* 주요 데이터 필드:
* Season: 시즌 (연도)
* Date: 경기 날짜
* Opponent: 상대팀 이름
* Location: 경기 장소 (Home/Away)
* Tottenham Goals: 토트넘 득점
* Opponent Goals: 상대팀 득점
* Python 코드:

*import* requests

*import* pandas *as* pd

*# API 요청*

url = "https://api.football-data.org/v2/competitions/PL/matches"

headers = {"X-Auth-Token": "YOUR\_API\_KEY"}

response = requests.get(url, *headers*=headers)

matches = response.json()['matches']

*# 데이터프레임 변환*

df = pd.json\_normalize(matches)

df = df[df['homeTeam.name'] == 'Tottenham Hotspur'] *# 토트넘 경기 필터링*

df = df[['season.startDate', 'utcDate', 'awayTeam.name', 'score.fullTime.homeTeam', 'score.fullTime.awayTeam']]

df.columns = ['Season', 'Date', 'Opponent', 'Tottenham Goals', 'Opponent Goals']

*# CSV 파일로 저장*

df.to\_csv('tottenham\_matches.csv', *index*=False)

**2-2. 데이터베이스 설계 및 저장**

* + - MySQL 데이터베이스 테이블 생성

CREATE TABLE tottenham\_matches (

Season VARCHAR(10),

Date DATE,

Opponent VARCHAR(50),

Location VARCHAR(10),

TottenhamGoals INT,

OpponentGoals INT,

PRIMARY KEY (Season, Date, Opponent)

);

* Python으로 데이터 업로드

*from* sqlalchemy *import* create\_engine

*# 데이터베이스 연결*

engine = create\_engine('mysql+pymysql://root:password@localhost/football\_db')

*# 데이터 업로드*

df['Location'] = df.apply(lambda *row*: 'Home' *if* *row*['Opponent'] != 'Tottenham Hotspur' *else* 'Away', *axis*=1)

df.to\_sql(*name*='tottenham\_matches', *con*=engine, *if\_exists*='replace', *index*=False)

**2-3. 데이터 처리**

* API에서 누락된 필드와 결측값 처리
* 날짜 형식 변환 (ISO8601 → 표준 날짜 형식)
* goal\_difference: 득점 차이 계산
* Result: 경기 결과 분류 (Win, Draw, Lose)

df['goal\_difference'] = df['TottenhamGoals'] - df['OpponentGoals']

df['Result'] = df.apply(lambda *row*: 'Win' *if* *row*['TottenhamGoals'] > *row*['OpponentGoals']

*else* ('Lose' *if* *row*['TottenhamGoals'] < *row*['OpponentGoals'] *else* 'Draw'), *axis*=1)

**2-4. API 설계**

* /api/matches: JSON 형식으로 모든 경기 데이터를 반환

*from* flask *import* Flask, jsonify

*import* pandas *as* pd

app = Flask(\_\_name\_\_)

*# 데이터베이스 연결 및 쿼리 실행*

df = pd.read\_sql("SELECT \* FROM tottenham\_matches", *con*=engine)

@app.route('/api/matches', *methods*=['GET'])

def get\_matches():

*return* jsonify(df.to\_dict(*orient*="records"))

*if* \_\_name\_\_ == '\_\_main\_\_':

app.run(*debug*=True)

**2-5. 테스트 및 오류 처리**

* 데이터베이스 연결 오류 처리:

*try*:

engine = create\_engine('mysql+pymysql://root:password@localhost/football\_db')

conn = engine.connect()

*except* Exception *as* e:

print(f"Error connecting to the database: {e}")

* API 호출 실패 대비:

*try*:

response = requests.get(api\_url, *headers*=headers)

response.raise\_for\_status()

*except* requests.exceptions.RequestException *as* e:

print(f"API Request failed: {e}")

1. **프로젝트**

* 데이터베이스 구성:
  + 경기 데이터 100건 이상 처리 및 저장
  + MySQL 테이블 설계에 따른 데이터 무결성 확보
* 주요 통계
  + 총 승률: **67%**
  + 평균 득점 차이: **+1.2**
* 과정

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

그림 1: 스크린샷 2024-11-17 오후 7.25.31

스크린샷, 소프트웨어, 웹 페이지, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

그림 2: 스크린샷 2024-11-17 오후 7.26.34텍스트, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그림 3: 스크린샷 2024-11-17 오후 7.26.50텍스트, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그림 4: 스크린샷 2024-11-17 오후7.52.58

* API 호출 결과

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

1. **참고자료**

FootaballDataAPI: <https://www.football-data.org/>