DATABASE(COMP322005)

K_COVID19 TeamProject
1조 보고서

2019113711 김현수
2019110944 이승신
2019113772 정명주

<목차>

I. 실행환경

1. 실행환경

A. OS : Window
B. Python version 3.8.2
C. MySQL version 5.7.31 (by WampServer)
D. PHP version 7.3.21 (by WampServer)

2. 조건

A. 공통 조건

- python 라이브러리
  -import pymysql : pip install PyMySQL
  -import pandas : pip install pandas
  -import math, csv, sys -> 기본 내장 라이브러리 파일

- 연결 시 port 번호 : 3306

B. 파일 위치(/Team1)

- .py 파일

➔ /1주차 : parsing_case.py, parsing_region.py, parsing_weather.py,
➔ /2주차 : makeTimeAge.py, makeTimeGender.py, makeTimeProvince.py
➔ /4주차 : parsing_hospital.py

- .csv 파일

➔ /data : K_COVID19.csv, addtional_Timeinfo.csv, Hospital.csv, Region.csv

- .sql 파일

➔ /1주차 : K_COVID19.sql
➔ /2주차 : TimeAge.sql, TimeGender.sql, TimeProvince.sql
➔ /3주차/3 : real_time_confirmed.sql
➔ /4주차 : hospital.sql

- .php 파일

➔ /3주차/1 : case.php, region.php, weather.php
➔ /3주차/2 : case2.php, region2.php, weather2.php, patient2.php
➔ /3주차/3 : real_time_confirmed.php

C. 파일 실행 방법(Usage)

- .py 파일 : 파일의 **user, password만 수정** 후 python 파일이 있는 위치에서 실행

➔ python <파일명>.py

- .php 파일 : **php파일들은 모두 /wamp64/www 폴더에 넣어서 wamp으로 구동시킴.**

➔ **인터넷 브라우저 - localhost/<파일명>.php (예: localhost/case.php)**

II. 코드 구현 이유

1. 1주차 과제

A. K_COVID19 데이터베이스 생성(CREATE DATABASE K_COVID19)
- 한국의 COVID-19 확진자에 대한 분석 정보를 만들기 위하여 'K_COVID19' 데이터베이스를 생성함.

B. 'PatientInfo' table에 대한 sql file
- 한국에 있는 COVID-19 확진자에 대한 모든 기본사항에 대한 table을 CREATE함
- 'patient_id'를 primary key값으로 선정함.

C. 'Case' table에 대한 sql file
- 한국의 COVID-19 감염 경우에 대한 정보들에 대한 table을 CREATE함
- 'case_id'를 primary key값으로 선정함.

D. 'Region' table에 대한 sql file
- 한국의 지역 위치와 지역에 대한 통계적 정보들에 대한 table을 CREATE함
- 'region_code'를 primary key값으로 선정함.

E. 'Weather' table에 대한 sql file
- 한국의 지역의 날씨 정보들에 대한 table을 CREATE함
- 'region_code'와 'wdate'를 primary key값으로 선정함.
- 지역과 날짜에 대한 날씨가 명확히 구분되는 special한 data라고 생각하였음.

F. 각 table에 대한 parsing Python file
- 제공된 'parsing_patient.py' 와 'K_COVID19.csv' 를 기반으로 table 'case', 'region', 'weather'에 대한 parsing 파이썬파일을 구현함.
- 각 table에 대한 정보들을 넣기 위하여 'K_COVID19.csv' 파일을 이용하였음. csv파일을 open 및 read 해주었음. 각 table에 필요한 column들을 csv파일에서 뽑아내 list를 만들었고, 이후 enumerate를 이용하여 파일을 한 줄씩 읽어 중복되는 key가 있는지 list를 이용하여 확인하였음. 최종적으로 INSERT 쿼리문을 작성하여 해당 줄에 해당되는 데이터를 각 table에 삽입하였음.

2. 2주차 과제

A. 'TimeInfo' table 에 대한 설명
- 날짜에 따른 한국의 COVID-19 확진에 대한 증감 정보를 table을 CREATE함.
- 'date'를 primary key값으로 선정함.

B. 'TimeAge' table에 대한 sql file
- 날짜에 따른 한국의 COVID-19 확진에 대한 증감 정보를 나이별로 구분하는 table을 CREATE함.
- 그 날의 나이대 별 확진 증감을 확인할 수 있는 'a_date'를 primary key값으로 선정함.

C. ‘TimeGender’ table에 대한 sql file
- 날짜에 따른 한국의 COVID-19 확진에 대한 증감 정보를 성별로 구분하는 table을 CREATE함.
- 그 날의 성별에 따 확진 증감을 확인할 수 있는 ’g_date’를 primary key값으로 선정함.

D. ‘TimeProvince’ table에 대한 sql file
- 날짜에 따른 한국의 COVID-19 확진에 대한 증감 정보를 지역별로 구분하는 table을 CREATE함.

E. ‘TimeAge’ table에 대한 parsing Python file
- 각 table에 대한 정보들을 넣기 위하여 ‘addtional_Timeinfo.csv’파일과 ‘K_COVID19.csv’ 파일을 이용하였음. 최종적으로 INSERT 쿼리문을 작성하여 해당 줄에 해당되는 데이터를 각 table에 삽입하였음.
- 나이대는 0s~100s까지 0대부터 100대 나이까지 list를 나눔.
- 또한, 나이대 별로 confirmd_date와 relased_date를 저장할 수 있는 list를 생성하였음.
- 제공된 ‘addtional_Timeinfo.csv’ file과 ‘K_COVID19.csv’ file을 모두 열어 날짜마다 확진 증감 정보와 나이대를 비교하고 해당하는 list에 append 해주었음.
- append한 정보를 각 날짜별 나이대에 따른 증감정보를 출력할 수 있도록 ‘TimeAge’ table에 INSERT 쿼리문을 작성하여 해당되는 데이터를 각 table에 삽입하였음.

F. ‘TimeGender’ table에 대한 parsing Python file
- 날짜와 성별에 따라 확진자, 사망자를 누적하여 저장하는 table이므로 날짜에 대한 정보가 필요하다고 판단, 날짜를 key로 하고 확진자 수를 value로 하는 cdate_dic_(fe)male 딕셔너리를 생성하였음. 해당 딕셔너리는 확진날짜에 대한 리스트를 탐색하며 male인경우, female인 경우를 구별하여 값을 + 1씩 해주었음. 해당 과정은 확진일, 사망일을 똑같이 진행해주었음.
- 이렇게 구한 날짜별 숫자를 누적해주기 위하여 total_confirmed(decreased)_(fe)male 변수에 더해가면서 저장해주었으며, 누적 날짜를 저장해주는 리스트를 이용하였음.
- INSERT 쿼리를 이용하여 해당 정보를 TimeGender table에 삽입해주었음.

G. ‘TimeProvince’ table에 대한 parsing Python file
- 날짜와 지역에 따른 누적 확진/완치/사망자수를 세야하므로 우선 ‘K_COVID19.csv’ 파일에서 province만 따로 추출함.
- 아래의 사전을 정의하여 추출한 province별 날짜에 따른 확진/완치/사망자수를 셈.
- pro_dic = { ‘province’ : [cdate_dic, rdate_dic, ddate_dic] }
- 아래의 사전을 정의하여 province별 누적 확진/완치/사망자를 셈.
- total = { ‘province’ : [total_confirmed, total_released, total_deceased] }
- ‘addtional_Timeinfo.csv’파일의 날짜에 대하여 pro_dic의 cdate_dic, rdate_dic, ddate_dic 날짜와 일치하는 경우 값을 누적시켜주었음.
- INSERT 쿼리를 이용하여 해당 정보를 TimeProvince table에 삽입해주었음.

3. 3주차 과제

A. 구현이유

a. 3개 table에 관한 php
- 먼저 th tag를 통해 각 attribute의 이름을 출력해주었음.
- select 쿼리문을 통하여 테이블의 정보를 가져왔음.
- 쿼리문을 통한 결과를 한 줄씩 읽으며 테이블에 출력해주었음.

b. 4개 table에 관한 select php(공통 구현 사항)
- 검색 방법을 post로 구현하였으며, 하나의 php파일을 이용하기 위하여 _POST['id'] 전역변수에 값이 존재하지 않는 경우(즉, 최초 페이지 실행 시) _POST['id']을 값으로 받는 $state1을 *로 설정하여 전체 정보를 출력 가능하도록 설정하였음. _POST['id']에 정보가 존재하는 경우 $state1을 해당 정보로 초기화해주었음.
- 사용자가 직접 어떤 정보를 검색하는 방식보다는 table에 있는 정보 중 선택하여 검색하는 방식이 더 편리하다고 판단하였음. 따라서 DB에 있는 정보가 옵션에 있는 dropbox 방식을 선택하게 되었음. 이러한 방식을 구현하기 위하여 SELECT tag를 이용하며, 실제 DB내의 정보를 가져오기 위해 SELECT 쿼리문을 이용하였음. 이렇게 가져온 정보는 checking array를 통하여 중복 체크 후 중복되지 않도록 option value로 추가해주었음.
- 옵션 선택 이후 input 버튼인 submit 선택 시 정보를 "<table명>.php"에 POST 하도록 하였음.

c. Patientinfo select php
- patient가 거주하는 city를 선정 : patient의 city에서 발생한 정보를 날짜 별로 확인하며 추가적 지역발생에 유의할 수 있음.

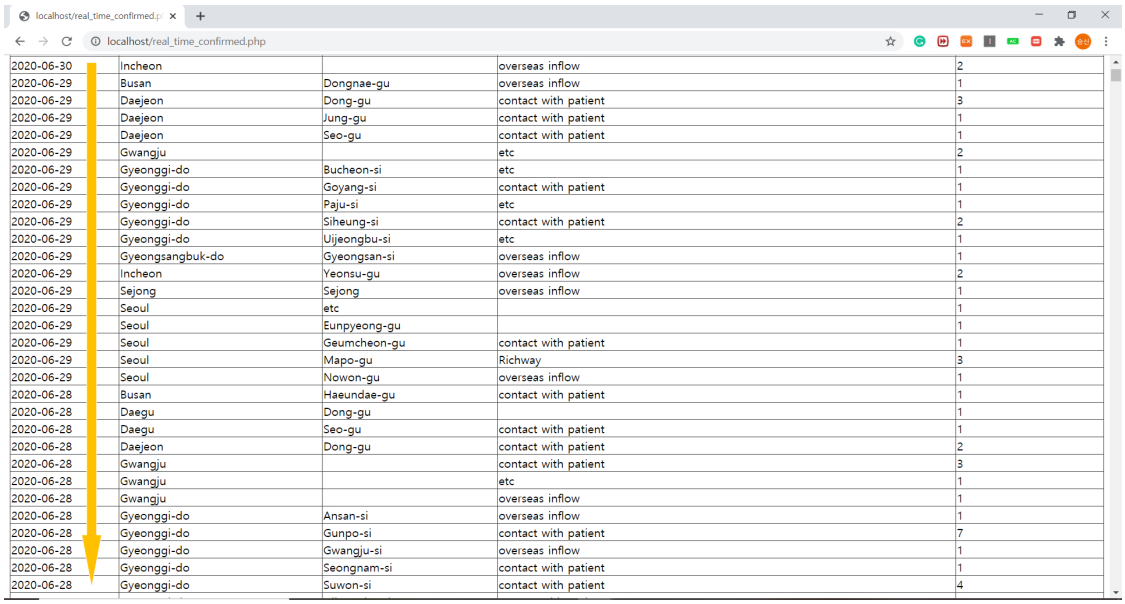
d. Case select php
- case의 infection_case를 선정 : infection_case별 data를 본다면 각각case별 감염정보에 대하여 이해도가 더 높아져 추후 예방책을 강구할 수 있음.

e. Region select php
- 해당 Region의 city를 선정 : Region별로 data를 보았을 때 가장 중요하고 사람들에게 필요한 attribute라고 생각하여 선정하였음.

f. Weather select php
- Weather의 날짜 wdate를 선정 : 우리가 주로 일별 확진자를 보듯이 일별 날씨에 따라 확진 추세를 확인할 수 있는 용이함이 있음.

g. real_time_confirmed View
- patientinfo의 4개의 attribute(confirmed_date, province, city , infection_case)와 이들을 grouping하여 count(*)를 통해 산출한 확진자수까지 총 5개의 attribute을 포함하고 confirmed_date를 기준으로 내림차순 정렬되는 view를 만듦.
- 최근날짜순으로 해당 날짜에 대한 지역/도시/사례별 확진자수를 한눈에 확인 가능.

B. 결과 설명



2020-06-30	Incheon		overseas inflow	2
2020-06-29	Busan	Dongnae-gu	overseas inflow	1
2020-06-29	Daejeon	Dong-gu	contact with patient	3
2020-06-29	Daejeon	Jung-gu	contact with patient	1
2020-06-29	Daejeon	Seo-gu	contact with patient	1
2020-06-29	Gwangju		etc	2
2020-06-29	Gyeonggi-do	Bucheon-si	etc	1
2020-06-29	Gyeonggi-do	Goyang-si	contact with patient	1
2020-06-29	Gyeonggi-do	Paju-si	etc	1
2020-06-29	Gyeonggi-do	Siheung-si	contact with patient	2
2020-06-29	Gyeonggi-do	Uijeongbu-si	etc	1
2020-06-29	Gyeongsangbuk-do	Gyeongsan-si	overseas inflow	1
2020-06-29	Incheon	Yeonsu-gu	overseas inflow	2
2020-06-29	Sejong	Sejong	overseas inflow	1
2020-06-29	Seoul	etc		1
2020-06-29	Seoul	Eunpyeong-gu		1
2020-06-29	Seoul	Geumcheon-gu	contact with patient	1
2020-06-29	Seoul	Mapo-gu	Richway	3
2020-06-29	Seoul	Nowon-gu	overseas inflow	1
2020-06-28	Busan	Haeundae-gu	contact with patient	1
2020-06-28	Daegu	Dong-gu		1
2020-06-28	Daegu	Seo-gu	contact with patient	1
2020-06-28	Daejeon	Dong-gu	contact with patient	2
2020-06-28	Gwangju		contact with patient	3
2020-06-28	Gwangju		etc	1
2020-06-28	Gwangju		overseas inflow	1
2020-06-28	Gyeonggi-do	Ansan-si	overseas inflow	1
2020-06-28	Gyeonggi-do	Gunpo-si	contact with patient	7
2020-06-28	Gyeonggi-do	Gwangju-si	overseas inflow	1
2020-06-28	Gyeonggi-do	Seongnam-si	contact with patient	1
2020-06-28	Gyeonggi-do	Suwon-si	contact with patient	4

real_time_confirmed에서 확진 날짜가 최신순으로 정렬되어 있음을 알 수 있으며, 해당 날짜에 같은 지역에서 같은 경로로 감염이 된 사람의 수를 알 수 있음.

4. 4주차 과제

A. 구현 이유

a. Hospital table sql file

- 과제 파일에 주어진 조건을 따라 Hospital table을 추가하기 위해 CREAT함
- patientinfo TABLE에 hospital_id attribute를 추가하기 위해 ALTER을 사용함

b. 조건을 만족하는 insert python file

- **hospital data 삽입 :**
  hospital table에 hospital.csv파일의 내용을 넣기 위하여 csv파일을 open 및 read 해주었음. 이후 enumerate를 이용하여 파일을 한 줄씩 읽어 중복되는 Hospital_id가 있는지 list를 이용하여 확인하였음. 최종적으로 INSERT 쿼리문을 작성하여 해당 줄을 K_COVID19 데이터의 hospital table에 삽입하였음.

- **patientinfo table의 hospital_id 값 삽입(환자 병원 배정):**
- **dup(list), Hospital_capacity(dict)={hid:[수용인원, 현인원] :**
  환자를 병원에 배정하기 위하여 가장 먼저 모든 hospital의 정보를 가지고 있어야 편하다고 판단, hospital.csv파일을 열어 모든 hospital 정보를 담은 dup list를 만들어주었음. 그리고 인원 관리의 편의성을 위하여 id, 수용인원, 현재인원 정보를 가져와야 한다고 판단, hid가 Key이고 [capacity, now]를 Value로 가지는 Hospital_capacity 딕셔너리를 만들어주었음.

- **hdis(dict)={(province, city):[가까운 순으로 정렬된 병원]:**
  Patientinfo에는 위도, 경도 정보는 존재하지 않으므로 Region.csv파일의 위도 경도를 이용해야 함. 그렇기에 미리 Region마다 가까운 병원이 어딘지 저장해두면 환자는 수용인원 정보만 판단하여 빠르게 병원을 선택할 수 있다고 생각했음.

따라서 Region.csv파일을 한 줄씩 읽어와 tuple(province, city)를 Key로 하고 가까운 순으로 정렬된 병원 리스트를 Value로 가지는 hdis 딕셔너리를 만들어주었음.

- **def near_hospital(지역정보) return [거리순으로 정렬된 병원]:**
  가까운 순으로 정렬된 병원 리스트를 생성하기 위하여 near_hospital이라는 함수를 정의해주었음. 해당 함수 내에서는 유클라디안 distance방식을 사용하여 현재 region의 위도, 경도와 모든 병원의 위도, 경도를 이용하여 distance를 구해주었음. 그리고 hid를 키로 하고 distance를 value로 하는 dis 딕셔너리를 생성하여 sorted함수를 이용, distance에 대하여 정렬하여 list를 생성하여 반환해주었음. 반환된 리스트는 앞서 설명한 hdis 딕셔너리의 Value로 사용됨.

- **patient_hos(dict)={환자id:병원id}:**
  환자에게 병원을 배정하기 위하여 환자 data가 필요하다고 판단, select 쿼리문을 이용하여 patientinfo table의 id, province, city 정보를 가져왔음. 가져온 튜플을 한 줄씩 읽어 hdis의 key인 province, city와 같은 경우를 찾음. (예외적으로 환자의 city정보가 None 혹은 etc인 경우 city 정보를 province정보와 같게 해 대표값을 사용할 수 있도록 해주었음.) 환자가 자신의 위치를 찾게 된다면 hdis의 value인 가까운 순으로 정렬된 병원 정보를 탐색하며 탐색하고 있는 병원의 현재 인원이 수용 인원보다 적으면 해당 병원에 환자를 배정하고 탐색을 끝냄. 그렇지 않은 경우 다음 가까운 병원을 탐색하여 배정 가능한 병원을 찾을 때 까지 반복함. 환자가 배정받을 병원을 찾게 된다면 환자의 id를 key로 하고 병원의 id를 value로 가지는 patient_hos 딕셔너리에 저장하고 Hospital_capacity의 now를 + 1 해줌.

- **patientinfo, hospital table UPDATE:**
  UPDATE 쿼리문을 사용하여 patient_id가 patient_hos의 key에 해당하는 column에 hospital_id를 patient_hos의 값으로 저장함.
  또한 hospital table 역시 Hospital_capacity를 이용하여 현재 인원 정보를 UPDATE해줌.

  <patient와 region 파일의 표기가 달라 병원 배정을 하지 않은 지역>
  (patientinfo내용->region.csv내용)
  Daegu Dalsung-gun -> Dalseong-gun
  Daegu Kyeongsan-si -> Gyeongsangbuk-do Gyeongsan-si
  Daegu sankyeock-dong -> X
  Daegu Yeongcheon-si -> Gyeongsangbuk-do Yeongcheon-si
  Daejeon Sejong -> Sejong Sejong
  Gyeonggi-do Yangpyeong-si
  Gyeonggi-do Suwon -> Suwon-si
  Gyeonggi-do Guri -> Guri-si
  Jeollanam-do Gyeongsan-si -> Gyeongsangbuk-do Gyeongsan-si

c. web을 구현한 php
  : 1조에서는 Hospital_id 입력 시 해당 병원에 입원한 환자의 정보를 출력하도록 페이지를 구현하였음.
- 검색 방법을 post로 구현하였으며, 하나의 php파일을 이용하기 위하여 _POST['id'] 전역변수에 값이 존재하지 않는 경우(즉, 최초 페이지 실행 시) _POST['id']을 값으로

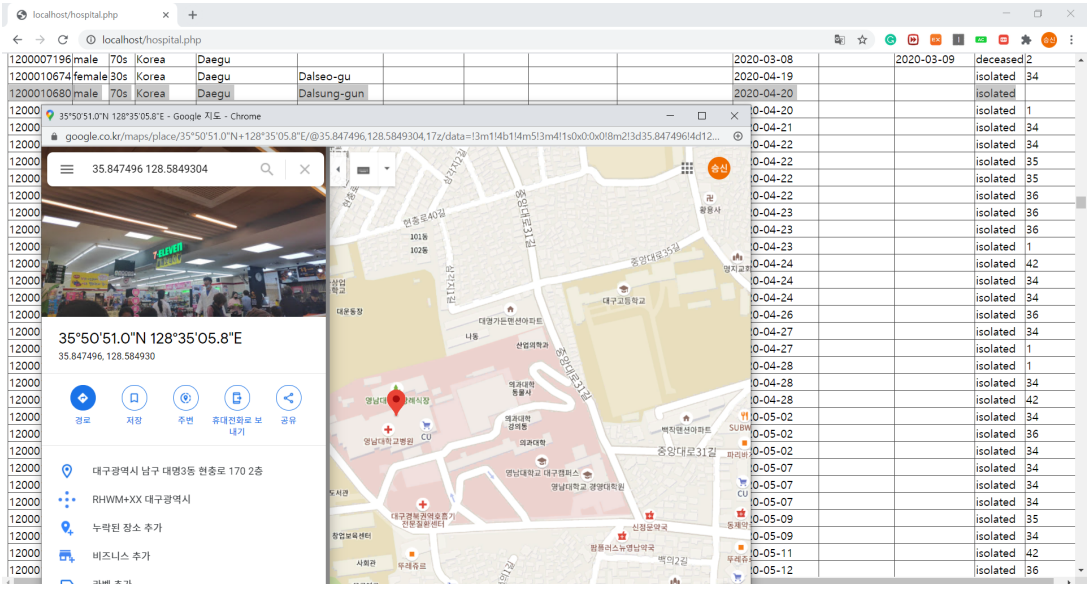
받는 $state1을 *로 설정하여 전체 정보를 출력 가능하도록 설정하였음.
_POST['id']에 정보가 존재하는 경우 $state1을 해당 정보로 초기화해주었음.

- $state1 변수의 정보에 따라 쿼리문을 바꾸어주었음. *인 경우는 patientinfo의 전체 정보를, *이 아닌 경우는 hid가 $state1의 값과 같은 정보만을 SELECT해주었음.
- mysqli_fetch_assoc 함수를 이용하여 쿼리 결과를 한 줄씩 읽어와 테이블로 출력해주었음.

d. php에 지도 연결

- 지도 연결을 위하여 병원의 name과 city를 이용하는 방법, 위도, 경도를 이용하는 방법 중 csv파일에 명확히 제공된 위도, 경도를 이용하는 방법을 택하기로 결정하였음.
- JavaScript를 이용하여 openInNewTab함수를 정의해주었음. 해당 함수는 hospital의 위도 경도를 받아서 google map의 url에 search 위도+ 경도를 붙여 검색된 페이지를 팝업창으로 나타나도록 함수를 구현하였음.
- 병원의 위도 및 경도를 알기 위하여 병원의 id, latitude, longitude 정보를 가져오는 SELECT 쿼리를 사용하였음. 해당 쿼리의 결과를 hosparry 연관 배열에 id를 key로, array(위도, 경도)를 value로 가지도록 저장하였음.
- 테이블 출력 과정에서 환자 정보 출력 시 현재 출력하는 것이 hospital_id인 경우 hosparray의 키가 같은지 비교하여 위도 경도를 가져와 openInNewTab 함수의 인자로 넘겨주었음. 현재 출력하는 것이 hospital_id가 아닌 경우는 그냥 val만 td index로서 출력하게 해주었음.
- 만약 환자가 배정받은 병원이 없는 경우(region과 환자 위치 이름이 맞지 않은 14개의 정보)에는 아무것도 연결되지 않게 해주었음.

B. 결과 설명



hospital.php 페이지에서 병원 번호를 누르면 해당 병원 위치에 해당하는 지도가 팝업창으로 나타남을 알 수 있음. 또한, 해당 팝업창은 구글맵에서 해당 병원의 위도, 경도 정보를 이용하여 위치를 search 한 것을 알 수 있음.
또한 환자 정보의 city(province)와 region.csv파일의 표기가 다른 예 중 하나인 달성군의 경우, 병원 배정이 이루어지지 않아 값이 없음을 확인 할 수 있음.