미국 중환자실 오픈데이터베이스 MIMIC-III

Tutorials

1. Tutorials

- sql-intro.md : 테이블과 중요한 SQL 키워드를 다루는 MIMIC-III 데이터베이스와 함께 SQL (Structured Query Language)을 사용하는 방법을 소개

2. sql-intro.md

- 관계형 데이터베이스에 대한 이해
- CSV 형식에 대한 이해
- MIMIC-III 데이터베이스에 대한 지식
- SQL (Structured Query Language)을 사용하여 데이터베이스에서 데이터를 선택하는 기능
- MIMIC 코드 리포지토리에서 코드를 재사용하는 기능
- SQL 집계 및 window 함수를 사용하는 기능
- 혈압 측정을 추출하는 기능

Comma separated value files(CSV)

- CSV (쉼표로 구분 된 값) 파일은 데이터를 표 형식의 스프레드 시트 스타일 구조로 저장하는 데 사용되는 일반 텍스트 형식
- 일반적인 권장 사항은 RFC 4180 사양 문서에서 Internet Engineering Task Force가 설정 한 CSV에 대한 정의
  - *파일은 선택적으로 헤더 행으로 시작할 수 있으며 각 필드는 쉼표로 구분됩니다.*
  - *레코드는 다음 행에 나열되어야합니다. 필드는 쉼표로 구분해야하며 각 행은 줄 바꿈으로 끝나야합니다.*
  - *숫자가 포함 된 필드는 선택적으로 큰 따옴표로 묶을 수 있습니다.*
  - *텍스트가 포함 된 필드("문자열")는 큰 따옴표로 묶어야합니다.*

관계형 데이터베이스

- 관계형 데이터베이스는 공유 키로 서로 연결된 테이블 모음으로 생각할 수 있습니다. 여러 테이블에 걸쳐 데이터를 구성하면 데이터 무결성을 유지하고 더 빠른 분석과보다 효율적인 스토리지를 구현.
- 관계형 데이터베이스가 필요한 이유?

한 사람에 대한 데이터 (이름, 나이 및 키)를 CSV 저장할때

"Name", "Age", "Height"

"Penny", 30, 182

페니의 심장 박동수를 4 시간 동안 오전 8시, 오전 9시, 오전 10시, 오전 11시에 측정값을 CSV로 저장할때

"Name", "Age", "Height", "Time", "Heart rate"

"Penny", 30, 182, "8:00", 65

"Penny", 30, 182, "9:00", 71

"Penny", 30, 182, "10:00", 72

"Penny", 30, 182, "11:00", 68

이 방식의 문제점은 무엇인가 ??

데이터베이스 용어

- "데이터베이스 스키마": 테이블의 구조와 관계를 정의하는 모델.
- "데이터베이스 쿼리": 구조화 된 "쿼리"를 사용하여 관계형 데이터베이스에서 데이터가 추출됩니다.
- "기본 키": 기본 키는 테이블의 각 행을 고유하게 식별하는 필드입니다.
- "외부 키": 외래 키는 다른 테이블의 기본 키를 나타내는 필드입니다.
- "정규화": 일반적으로 하나 이상의 테이블을 조인하도록하여 데이터 반복을 줄이고 데이터 무결성을 향상시키는 방식으로 데이터베이스를 구성하는 개념.
- "비정규 화": 때로는 데이터 반복 및 데이터 무결성을 희생하면서 가독성을 향상시키기 위해 데이터베이스를 구성하는 개념.
- "데이터 유형": 데이터의 동작 및 보유 할 수있는 가능한 값을 설명하는 데 사용되는 용어입니다 (예 : 정수, 텍스트 및 날짜는 모두 PostgreSQL의 데이터 유형입니다).

SQL (Structured Query Language)이란

- SQL (Structured Query Language)은 관계형 데이터베이스를 관리하는 데 사용되는 프로그래밍 언어입니다. SQL 쿼리는 다음 형식을 갖춤.

```
SELECT [columns]
FROM [table_name];
```

- 쿼리 결과는 일반적으로 관심있는 테이블에서 선택한 행 목록

```
SELECT subject_id
FROM patients;
```

- 문자는 모든 열을 선택하는 데 사용할 수 있는 와일드 카드

```
SELECT *
FROM patients;
```

WHERE 절

- 조건을 만족하는 데이터의 하위 집합을 조회할때 사용

```
SELECT [columns]
FROM [table_name]
WHERE [conditions];
```

- 여성 대상에 해당하는 모든 항목을 쉽게 조회

```
SELECT subject_id
FROM patients
WHERE gender = 'F';
```

- 단일 주제에 대한 모든 데이터를 조회

```
SELECT *
FROM patients
WHERE subject_id = 109;
```

WHERE 절

- WHERE절은 표준 논리 연산자와 결합 할 수 있습니다 AND/ OR

```
SELECT *
FROM patients
WHERE subject_id = 109
OR subject_id = 117
OR subject_id = 127;
```

- OR같은 열에있는 명령문 의 유용한 속기는 다음과 같은 IN조건

```
SELECT *
FROM patients
WHERE subject_id IN (109, 117, 127);
```

- "보다 작음"(<), "보다 작거나 같음" <=, "보다 큼"(>) 또는 "보다 크거나 같음" >=연산자

```
SELECT *
FROM patients
WHERE subject_id >= 109
AND subject_id <= 127;
```

WHERE 절

- SQL은 조건(>=)및 조건(<=) 과의 조합을 제공 (BETWEEN)

```
SELECT *
FROM patients
WHERE subject_id BETWEEN 109 AND 127;
```

- 텍스트 데이터로 작업 할 때 종종 정확한 일치가 아닌 부분 문자열 일치를 검색(LIKE)

```
-- use `LIKE` to match text
-- The `%` is a wildcard that will match all characters
SELECT *
FROM icustays
WHERE first_careunit LIKE '%ICU%';
```

ORDER BY 절

- 특정 컬럼으로 데이터를 정렬

```
SELECT [columns]
FROM [table_name]
WHERE [conditions]
ORDER BY [columns];
```

- Dob 컬럼으로 환자의 결과를 정렬

```
SELECT subject_id, dob
FROM patients
ORDER BY dob;
```

- WHERE절은 선택 사항이며 위 쿼리에서 생략 가능. 그러나 키워드의 순서를 준수 필요.

ORDER BY 절

- 특정 컬럼으로 데이터를 정렬

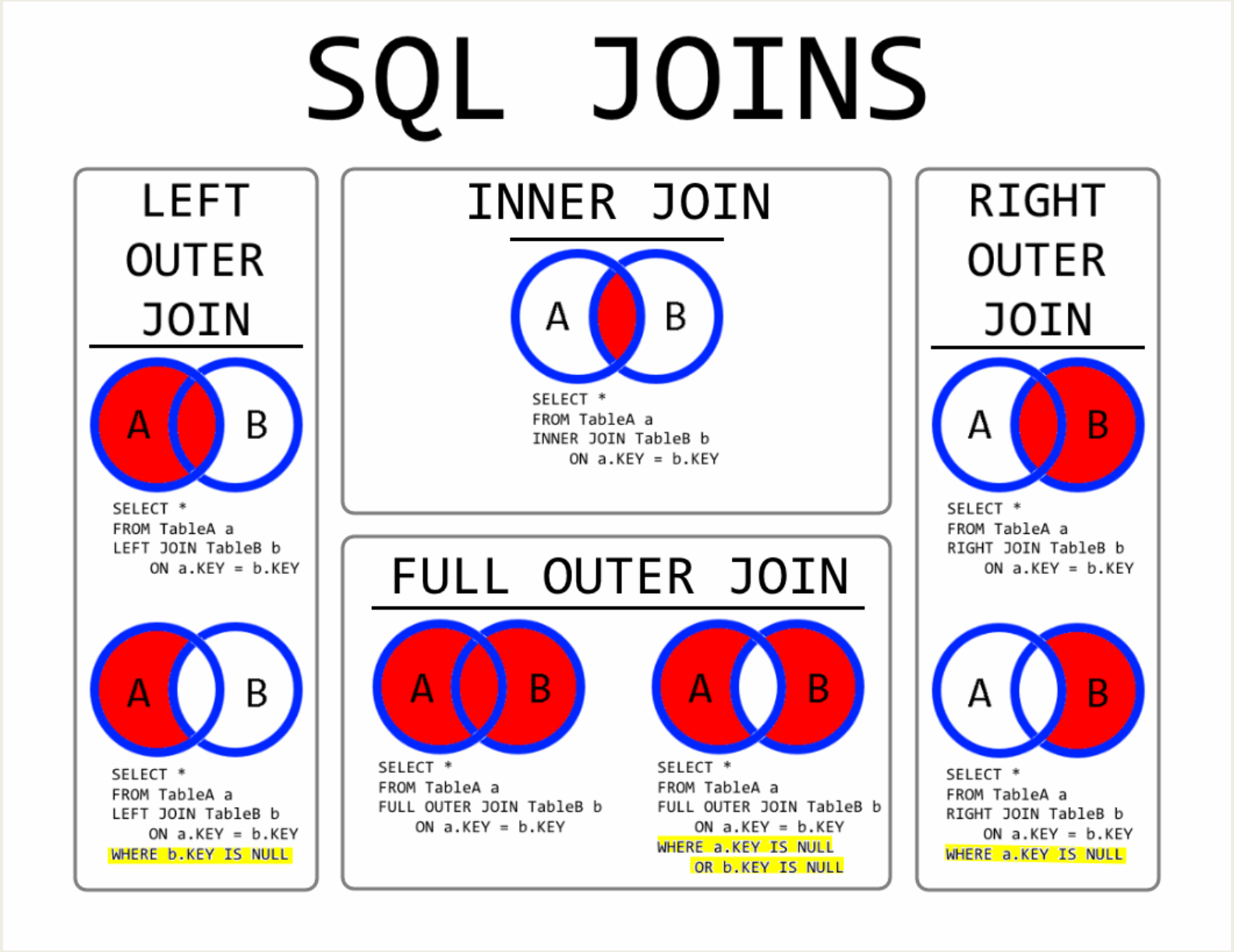
```
SELECT [columns]
FROM [table_name]
WHERE [conditions]
ORDER BY [columns];
```

- Dob 컬럼으로 환자의 결과를 정렬

```
SELECT subject_id, dob
FROM patients
ORDER BY dob;
```

- WHERE절은 선택 사항이며 위 쿼리에서 생략 가능. 그러나 키워드의 순서를 준수 필요.

SQL JOIN : 여러 개의 테이블을 조합하기



SQL JOIN

- patients 테이블에서 생년월일과 admission 테이블에서 입원 기간을 조회하는 쿼리
- 이 2개의 테이블을 결합할때 subject_id 컬럼을 키로 사용함.

```
-- INNER JOIN will only return rows where subject_id
-- appears in both the patients table and the admissions table
SELECT p.subject_id, p.dob, a.hadm_id, a.admittime
FROM patients p
INNER JOIN admissions a
ON p.subject_id = a.subject_id
ORDER BY subject_id, hadm_id;
```

컬럼에 대한 연산 함수들

- 컬럼의 데이터를 그대로 사용할 수 있고, 엑셀과 같이 데이터에 대한 변환이 가능.
- 예를 들어, los가장 가까운 날까지 체류 기간 ( ) 에만 관심이 있다면 다음 round함수를 사용

```
SELECT icustay_id, round(los)
FROM icustays;
```

- 수학 연산자를 사용해서 2개 이상의 컬럼에 대해서 작업이 가능.
- 예를 들어, 사망 한 환자의 병원 체류 시간 계산에 관심

```
-- When combining columns in an operation, it is sometimes necessary
-- to convert ('cast') them to the same data type
SELECT subject_id, admittime, deathtime
  , deathtime - admittime AS length_of_stay
FROM admissions
WHERE deathtime IS NOT NULL;
```

- IS NOT NULL. 값이 null이 아닌지 확인하는 절입니다 ("null"은 빈 값이며 누락 된 데이터를 나타냄).

쿼리 관리를 위한 임시 테이블

- 큰 쿼리를보다 작고 관리하기 쉬운 덩어리로 나누기 위해 임시 뷰나 테이블을 만드는 것이 도움
- WITH키워드를 사용
- 예를 들어 나이를 계산해놓은 patient_dates 이란 이전 쿼리를 임시 뷰로 만든 다음 모든 열을 선택

```
WITH patient_dates AS (
SELECT p.subject_id, p.dob, a.hadm_id, a.admittime,
    ( (cast(a.admittime as date) - cast(p.dob as date)) / 365.2 ) as age
FROM patients p
INNER JOIN admissions a
ON p.subject_id = a.subject_id
ORDER BY subject_id, hadm_id
)
SELECT *
FROM patient_dates;
```

- 다른 방법은 데이터베이스 스키마에 새 테이블을 만드는 "구체화 된 뷰“, 데이터베이스 테이블로 취급

```
DROP MATERIALIZED VIEW IF EXISTS patient_dates_view;

CREATE MATERIALIZED VIEW patient_dates_view AS

SELECT p.subject_id, p.dob, a.hadm_id, a.admittime,

    ( (cast(a.admittime as date) - cast(p.dob as date)) / 365.2 ) as age

FROM patients p

INNER JOIN admissions a

ON p.subject_id = a.subject_id

ORDER BY subject_id, hadm_id;
```

if/else 로직을 위한 CASE 문

- CASE명령문은 if / else 논리를 처리하는 데 사용
- 예를 들어, icustays 테이블을 사용하면 ICU 체류 길이 (los)를 짧게, 중간, 길게 그룹

```
SELECT subject_id, hadm_id, icustay_id, los,
   CASE WHEN los < 2 THEN 'short'
        WHEN los >=2 AND los < 7 THEN 'medium'
        WHEN los >=7 THEN 'long'
        ELSE NULL END AS los_group
FROM icustays;
```

- patients 테이블에서 gender 컬럼을 가지고 남자와 여자를 1 / 0 으로 코드화

```
SELECT subject_id, gender
, CASE WHEN gender = 'M' then 1
      WHEN gender = 'F' then 0
  ELSE NULL END
  as gender_binary
FROM patients;
```

집계 함수들

- 환자 수, 평균 심박수 또는 최대 혈압과 같은 여러 행에서 값을 집계할때는 COUNT(), MAX(), SUM(), AVG() 함수를 사용
- icustays 테이블 의 행 수를 계산 (count)

```
SELECT count(*)
FROM icustays;
```

- icustays 테이블에서 채류 기간이 가장 긴 일수 찾을때 (max)

```
SELECT MAX(los)
FROM icustays;
```

- GROUP BY 절을 사용하여 환자별 채류 기간일 가장 긴 일수를 추출

```
SELECT subject_id, MAX(los)
FROM icustays
GROUP BY subject_id;
```

집계 함수들

- WHERE절은 집계 열을 필터링하지 않으므로 HAVING키워드를 사용
- 예를 들어, 각 환자별로 그룹화 된 최대 체류 기간을 찾을 수 있으며 최대 체류 기간이 10 일 미만인 환자만을 조회

```
-- find the maximum length of stay in the ICU
-- for each patient
-- where the maximum length of stay is < 10 days
SELECT subject_id, MAX(los)
FROM icustays
GROUP BY subject_id
HAVING MAX(los) <= 10;
```

window 함수들

- 예를 들어, 각 환자의 ICU 입학 순서를 나열하는 컬럼을 생성하고 싶을때
- subject_id로 GROUP BY로는 가능하지 않고 RANK() window 함수를 사용하여 환자의 ICU 입원 순서를 조회할 수 있음.

```
SELECT subject_id, icustay_id, intime,
    RANK() OVER (PARTITION BY subject_id ORDER BY intime)
FROM icustays;
```

	subject_id	icustay_id	intime	rank
1	2	243,653	2138-07-17 21:20:07	1
2	3	211,552	2101-10-20 19:10:11	1
3	4	294,638	2191-03-16 00:29:31	1
4	5	214,757	2103-02-02 06:04:24	1
5	6	228,232	2175-05-30 21:30:54	1
6	7	278,444	2121-05-23 15:35:29	1
7	7	236,754	2121-05-25 03:26:01	2
8	8	262,299	2117-11-20 12:36:10	1
9	9	220,597	2149-11-09 13:07:02	1
10	10	288,409	2103-06-28 11:39:05	1
11	11	229,441	2178-04-16 06:19:32	1

- 임시 테이블과 결합하여 각 환자의 첫 번째 ICU 체류 만 선택

```
WITH icustayorder AS (
SELECT subject_id, icustay_id, intime,
  RANK() OVER (PARTITION BY subject_id ORDER BY intime)
FROM icustays
)
SELECT *
FROM icustayorder
WHERE rank = 1;
```