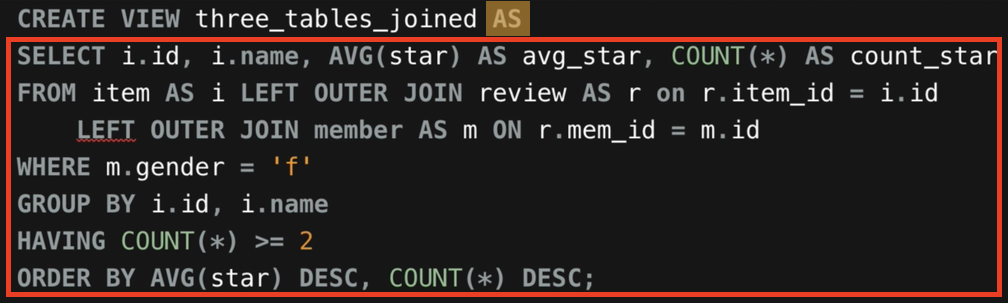
이전 영상에서 **뷰(View)**를 ‘**가상 테이블**’이라고도 한다고 말했습니다. 그럼 뷰와 그냥 테이블은 무슨 차이가 있는 걸까요?

가장 큰 차이는 뷰는 테이블과 달리 데이터가 물리적으로 컴퓨터에 저장되어 있는 건 아니라는 점입니다.

테이블은 우리가 표 형식으로 보는 데이터들이 실제로 컴퓨터에 저장되어 있습니다.

그런데 뷰는 표 형식으로 내용을 본다는 점에서는 테이블과 같지만, 테이블처럼 그 내용이 실제로 컴퓨터에 일일이 저장되어 있는 건 아니라는 점이 다릅니다. 그 대신 뷰는, 우리가 뷰를 사용할 때, DBMS가 그 뷰를 생성하는 SQL 문을 재실행하는 방식으로 가상의 테이블을 만들어주는 겁니다. 잠깐 아래 그림으로 설명해보자면,



우리가 three\_tables\_joined 라는 뷰를 사용할 때마다 AS 다음에 있는 빨간 박스 안의 SELECT 문이 재실행된다는 뜻입니다.

즉, 뷰라는 건, 테이블처럼 컴퓨터에서 데이터 크기만큼의 물리적인 용량을 차지하고 있는 것은 아니라는 뜻입니다. (요즘에는 자주 사용하는 뷰인 경우, 테이블처럼 데이터가 물리적으로 저장되도록 하는 기능도 있는데요. 중요한 내용은 아니니까 넘어갈게요)

뷰는 데이터베이스에 저장된 데이터를 분석해야할 때 매우 유용한 개념이자 기능입니다. 뷰의 장점들을 하나씩 살펴보겠습니다.

**첫 번째, 뷰는 사용자에게 높은 편의성을 제공해줍니다.** 이전 영상에서 본 것처럼 여러 테이블을 조인하는 SQL문을 매번 필요할 때마다 사용하는 것은 정말 번거로운 일입니다. 하지만 이런 복잡한 SQL문을 뷰로 한번 저장해두면 계속 재활용할 수 있어서 정말 편리하죠. 이건 이미 실감해봤으니까 바로 넘어갈게요.

**두 번째, 각 직무별 데이터 수요에 알맞은, 다양한 구조의 데이터 분석 기반을 구축해둘 수 있습니다.** 같은 테이블들이 존재하는 상황에서도, 직무에 따라, 상황에 따라, 필요로 하는 데이터의 종류와 그 구조가 사람마다 다를 수 있는데요. 뷰를 사용하면 각자에게 적합한 구조로 데이터들을 준비해둘 수 있기 때문에 회사 입장에서도 기존의 테이블 구조를 건드리지 않고, 풍부한 데이터 분석 기반을 준비할 수 있게 됩니다.

**세 번째, 뷰는 데이터 보안을 제공합니다.** 이 부분이 상당히 중요한데요. 예를 들어, 회사에서 직원들에 관한 정보를 담고 있는 **employee**라는 테이블이 있고 이 테이블에는 굉장히 민감한 정보가 담긴 컬럼들이 있다고 가정합시다. 예를 들어, **주민등록번호**나 **주소**, **연봉** 등과 같은 정보 말이죠. 그런데 회사 내의 데이터 분석가가 어떤 분석을 하기 위해 이 **employee** 테이블이 필요할 수도 있습니다. 하지만 아무리 데이터 분석을 해야한다고 해도 중요한 정보를 분석가가 마음대로 볼 수 있게 하는 것은 옳지 않습니다. 이때 분석가에게 민감 정보가 담긴 컬럼을 제외하고 보여줄 수 있는 방법도 바로 뷰입니다.

예를 들어, **employee** 테이블에 **id, name, age, department, registration\_number(주민등록번호), annual\_salary(연봉)** 컬럼이 있다고 가정했을 때, 이 테이블을 바로 데이터 분석가가 사용할 수 있도록 하면 안 됩니다.

이런 상황에서는

**CREATE VIEW emp\_view AS SELECT id, name, age, department FROM employee;**

같은 SQL 문을 실행해서 테이블과 비슷한 이름의 뷰를 만드는 게 좋습니다.

특정 column들 뿐만 아니라 특정 row들을 공개하지 않는 것도 가능합니다. 예를 들어 **department**의 값이 '**secret**'인 직원들의 정보를 공개하지 않아야 한다고 가정해본다면, 이런 뷰를 만들면 되겠죠?

**CREATE VIEW emp\_view2 AS SELECT id, name, age, department FROM employee WHERE department != 'secret';**

이렇게 WHERE 절로 조건을 붙여서, 특정 row들만 보여주는 뷰를 만드는 것도 가능합니다.

이런 SQL 문을 사용해서, 공개 가능한 정보만 있는 뷰를 만들고 난 후에는,

데이터 분석가가 **employee** 테이블에 직접적인 접근을 하지 못하도록 막고(DBMS에서 '사용자별 권한 관리' 기능을 통해 할 수 있습니다), **emp\_view 뷰에만 접근할 수 있도록 하면 됩니다.**

나중에 여러분이 회사에서 데이터 분석을 해야할 때는 테이블 자체보다는 이러한 뷰를 제공받을 가능성이 더 높습니다. 실제 서비스에서 사용되는 테이블에는 보통, 아무나 접근해서 작업을 할 수 없도록 보안 정책이 철저하게 구축되어 있을 겁니다. 하지만 그렇다고 그런 권한을 가진 사람이 데이터 분석 직무를 맡게 되는 것도 아니죠. 바로 이런 현실적인 문제를 해결하는 수단이 뷰가 되는 겁니다. 뷰가 어떻게 쓰이는지 감이 오시죠?

혹시 나중에 실무에서 '테이블을 직접 다루는 게 아니구나'라고 실망하지 마세요. 만약 제공받은 뷰로 부족함을 느낀다면 테이블에 직접 접근할 수 있는 개발자 분 등에게 요청해서 원하는 구조의 뷰를 제공받으면 됩니다.