이전 노트에서는 서브쿼리가 어떤 형식(column, row 수 기준)의 결과를 리턴하는지에 따라 그 종류를 나눠봤습니다.

그런데 서브쿼리를 그 리턴 결과가 아닌 다른 측면에서도 분류해볼 수 있는데요.

서브쿼리를 다른 방식으로 분류하는 방법은,

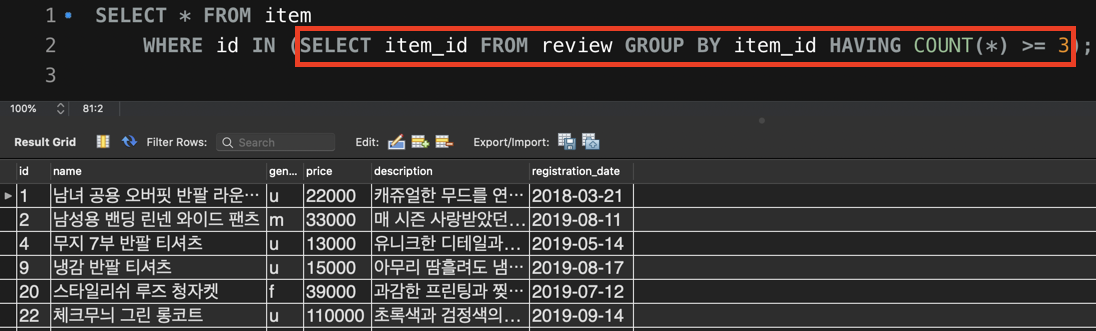
서브쿼리를

(1) **비상관 서브쿼리**와

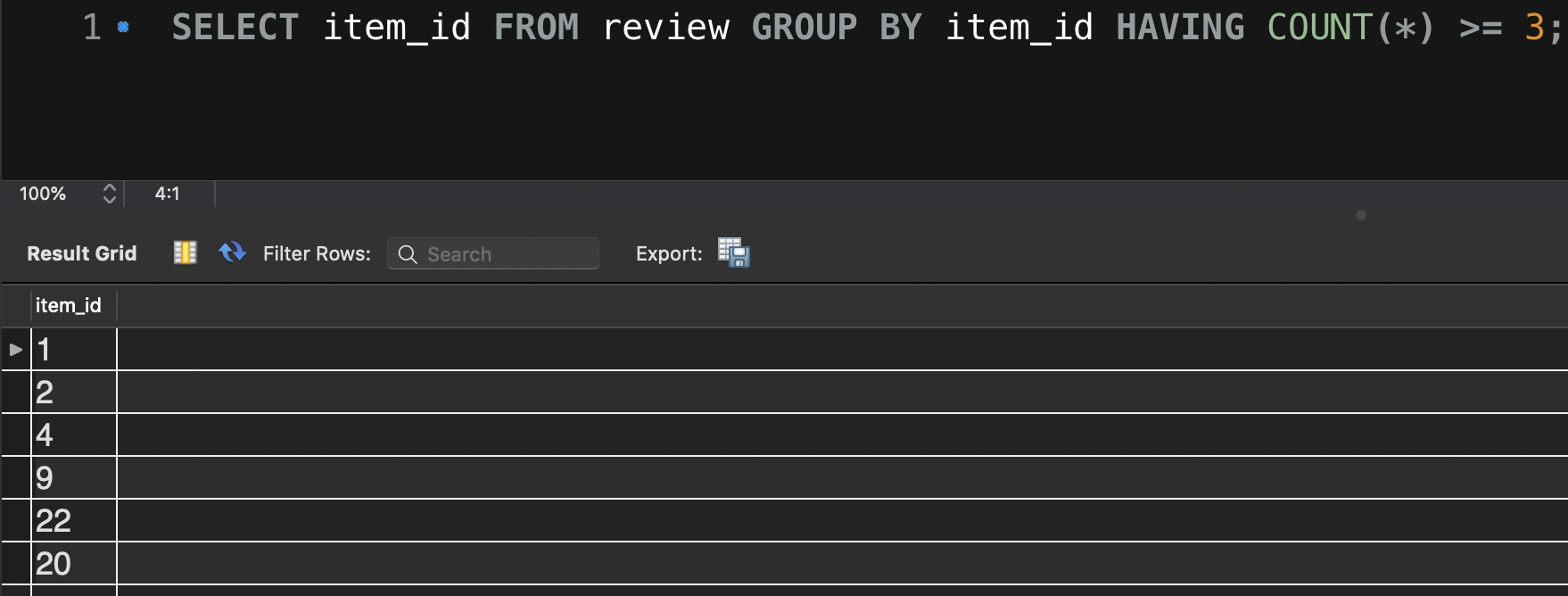
(2) **상관 서브쿼리**로

분류하는 것입니다. 이 내용은 좀 어려우니까 집중해서 읽어보세요.

비상관 서브쿼리가 뭘까요? 잠깐 이전 영상에서 배운 SQL 문을 보고 설명해드릴게요.

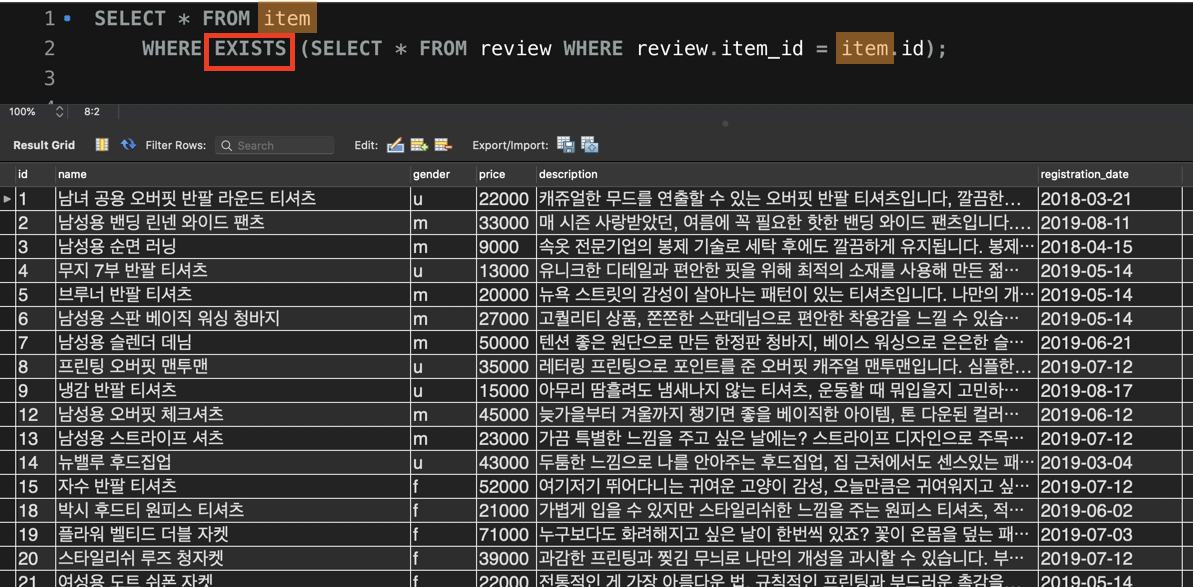


위 그림을 보시면 WHERE 절에서 서브쿼리가 사용되고 있는데요. 이 서브쿼리는 지금 **그 자체만으로도 실행이 가능한 서브쿼리**입니다. 따라서 이 서브쿼리만 빼서 아래 그림처럼 별도로 실행을 해봐도



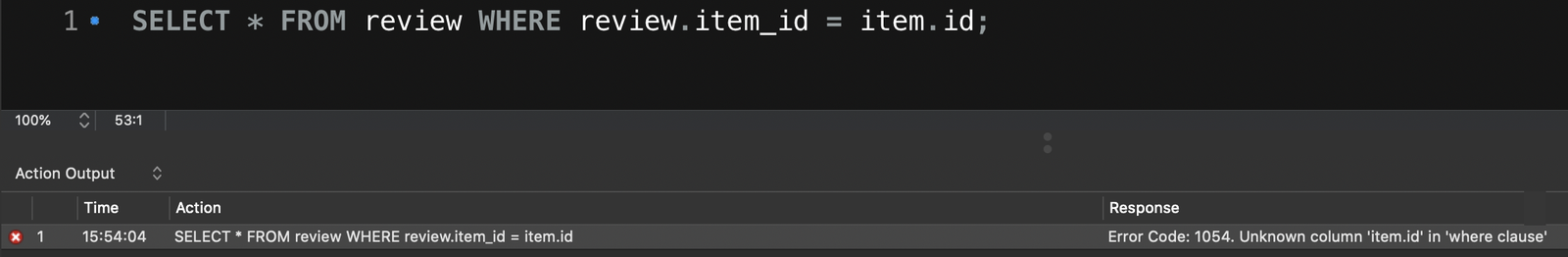
잘 실행됩니다. 이것은 **이 서브쿼리가 그것을 둘러싼 outer query와 별개로, 독립적으로 실행되기 때문에** 그런 겁니다. 이렇게 outer query와 상관 관계가 없는 서브쿼리를 **비상관 서브쿼리**라고 합니다. **이때까지 우리가 배운 서브쿼리들이 모두 비상관 서브쿼리에 해당합니다.**

그렇다면 상관 서브쿼리는 무엇일까요? 상관 서브쿼리는 우리가 지금 새롭게 배워야할 종류의 서브쿼리입니다. **상관 서브쿼리란 outer query와 상관 관계가 있는 서브쿼리를 말하는데요.** 아래 예시와 함께 설명할게요.



지금 이 그림을 보면 WHERE 절에 서브쿼리가 하나 쓰였습니다. 그리고 그 앞에 EXISTS라는 처음 보는 단어가 있는데요. EXISTS의 의미는 곧 바로 설명할게요. 일단 지금 서브쿼리의 뒷 부분을 보면 **item**이라는 테이블 이름이 있다는 것을 알 수 있습니다. 그런데 여기서 **신기한 사실은 item 테이블의 이름이 서브쿼리의 FROM 절에 있는 게 아니라 outer query에 있다는 점인데요. 주황색 강조 처리된 두 개의 item이라는 테이블 이름을 보세요.**

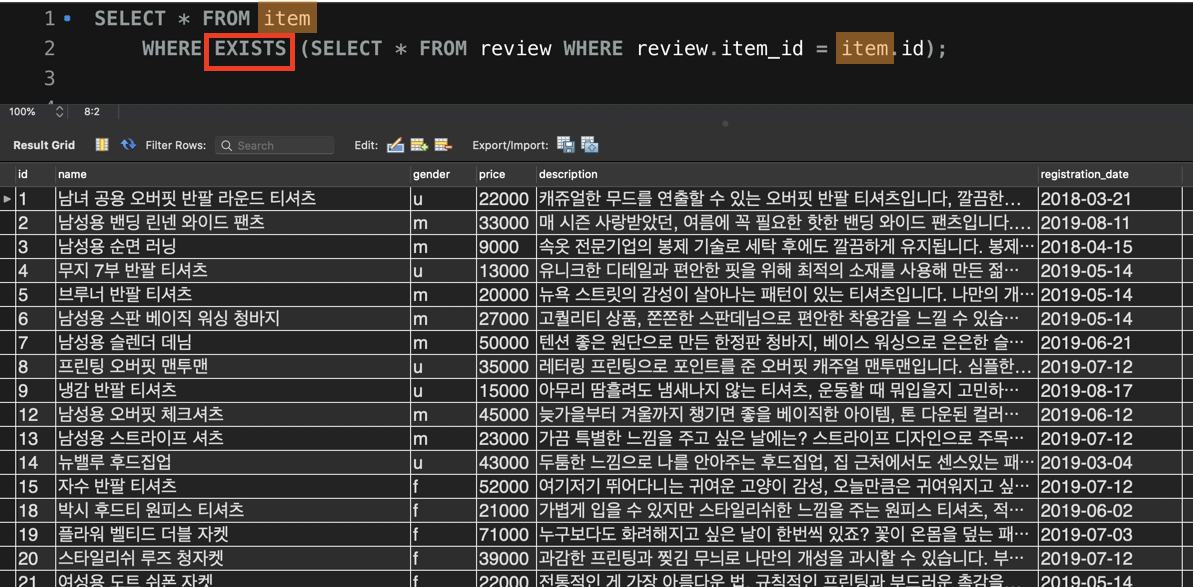
지금 서브쿼리가 필요로 하는 item 테이블이 outer query에 적혀있기 때문에 이 서브쿼리는 방금 전 '비상관 서브쿼리' 때와는 달리 단독으로 실행되지 못합니다. 잠깐 이 서브쿼리만 별도로 빼서 실행해보면,



**방금 전의 비상관 서브쿼리와는 달리 실행에 실패합니다.** item이라는 테이블 이름이 지금 FROM 절에 없기 때문에 당연한 겁니다.

이렇게 **서브쿼리가 outer query에 적힌 테이블 이름 등과 상관 관계를 갖고 있어서 그 단독으로는 실행되지 못하는 서브쿼리를 상관 서브쿼리라고 합니다.**

그렇다면 지금



이 SQL 문 전체는 무슨 뜻일까요? 하나씩 해석해볼게요.

이 SQL 문의 최종 조회 결과를 하나의 바구니라고 생각해봅시다.

그리고 시작할게요.

(1) 일단 item 테이블의 첫 번째 row를 생각합시다.

(2) 그 row의 id(**item.id**) 값과 같은 값을 item\_id(**review.item\_id**) 컬럼에 가진 review 테이블의 row(가/들이) 있는지 조회합니다.

(3) 만약에 존재하면(EXISTS의 의미가 바로 이것입니다!, exist는 우리말로 '존재하다'라는 뜻입니다)

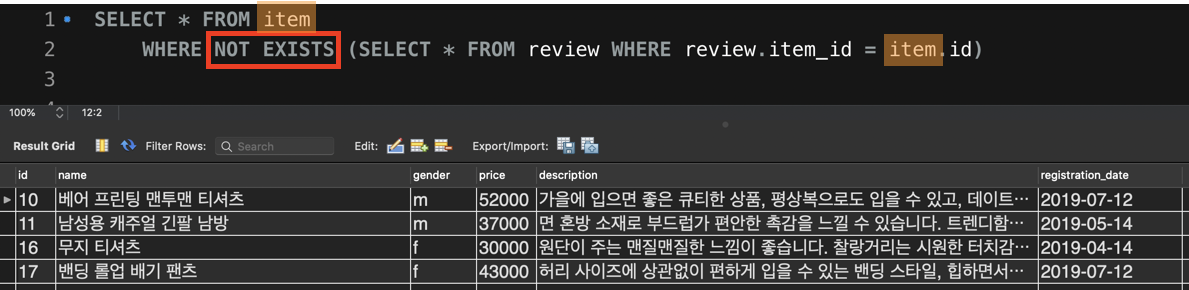
(4) WHERE 절은 True가 되고, (1)에서 생각했던 item 테이블의 row는 최종 조회 결과 바구니에 담깁니다.

이제 item 테이블의 두 번째 row에 대해서 (2) ~ (4)의 과정을 반복합니다. → 세 번째 row에 대해서 (2) ~ (4)의 과정을 반복합니다.

→ item 테이블의 마지막 row까지 (2) ~ (4)의 과정을 반복합니다.

이렇게 되면 **item 테이블 중에서 그 id 컬럼 값이 review 테이블의 item\_id 컬럼에 존재하는 row들만 추려지겠죠?** 이 말을 조금 바꿔서 말하면 **상품들 중에서 리뷰가 달린 상품들만 조회**한 것입니다. 상관 서브쿼리가 뭔지 조금 감이 오시나요?

이것과 정 반대로 아직 리뷰가 달리지 않은 상품들만 조회하는 방법도 있습니다. 아래 그림과 같이 EXISTS 대신 **NOT EXISTS**를 사용하면 됩니다.



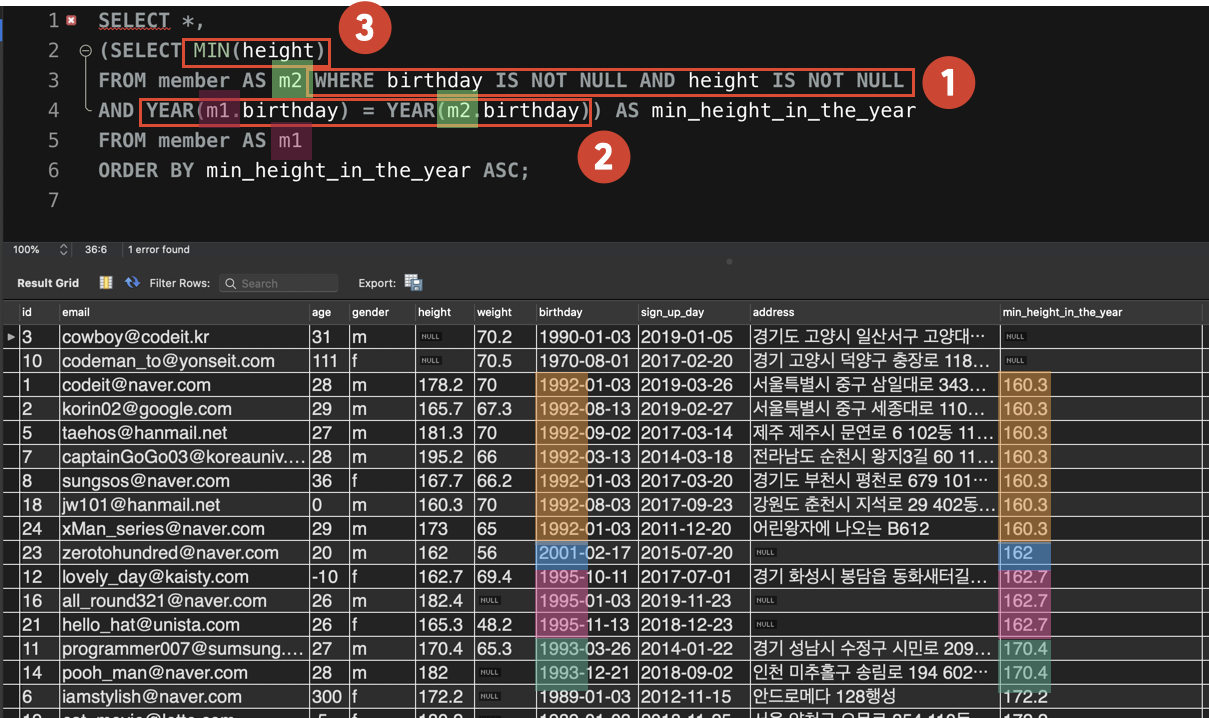
NOT EXISTS는 EXISTS와 정반대 의미로, 위에서 설명한 단계들에서 ‘(3)번 만약에 존재한다면’을 ‘**(3)번 만약에 존재하지 않는다면**’으로 바꾸고 생각하시면 됩니다. NOT EXISTS를 사용하니 리뷰가 달리지 않은 상품들만 잘 조회되었습니다.

상관 서브쿼리가 뭔지, 상관 서브쿼리에서 사용되는 EXISTS, NOT EXISTS의 의미가 뭔지 이해되시나요?

EXIST, NOT EXISTS는 상관 서브쿼리와 함께 자주 사용되는 키워드이기 때문에 그 의미를 잘 기억해두시는 게 좋습니다.

하지만 **상관 서브쿼리라고 해서 꼭 EXISTS, NOT EXISTS 같은 키워드를 써야만 하는 건 아닙니다.** 잠깐 아래 그림을 보세요. 상관 서브쿼리로 이런 것도 할 수 있습니다.

지금 저는 member 테이블을 조회하면서, 같은 해에 태어난 회원들 중 **가장 작은 키**를 가진 회원의 키 정보를 담은 컬럼을 오른쪽 끝에 추가해서 보려고 하는데요. 방금 전에 배운 상관 서브쿼리의 해석 방법대로 아래 SQL 문을 해석해보세요.



지금 보면 member 테이블 하나를 갖고 마치 이전에 배운 [SELF JOIN 같은 작업](https://www.codeit.kr/learn/3230)을 처리하고 있다는 걸 알 수 있습니다.

혹시 해석이 바로 안 되시는 분들을 위해 이미지에 있는 번호대로 하나씩 설명해드릴게요.

1번 : 일단 birthday 컬럼과 height 컬럼에 둘다 값이 있는 회원들만 대상으로 해야하기 때문에 이런 조건을 걸었습니다.

그 다음 member 테이블의 첫 번째 row를 생각합시다.

2번 : 그 row에 대해서 같은 **YEAR(birthday) ('생일연도')** 값을 가진 row(를/들을) 찾습니다.

3번 : 그 다음 해당 row(의/들의) height 컬럼의 최솟값을 구합니다.

member 테이블의 두 번째 row를 생각하고, 2~3번을 수행합니다.

member 테이블의 세 번째 row를 생각하고 ,2~3번을 수행합니다.

…

이런 식으로 구하다 보면 특정 회원과 같은 해에 태어난 사람들 중 가장 작은 키를 가진 사람의 키를 마지막 컬럼에서 볼 수 있습니다.

위 결과를 보면 같은 해에 태어난 회원들은 min\_height\_in\_the\_year라는 컬럼에서 모두 같은 값을 갖고 있는 것을 확인할 수 있습니다.

자, 상관 서브쿼리는 처음 배울 때는 조금 어려울 수도 있는 개념입니다. 위 내용을 이해될 때까지 반복해서 읽어보세요.

참고로 비상관 서브쿼리는 영어로 **Non-correlated Subquery**, 상관 서브쿼리는 영어로 **Correlated Subquery**라고 합니다.