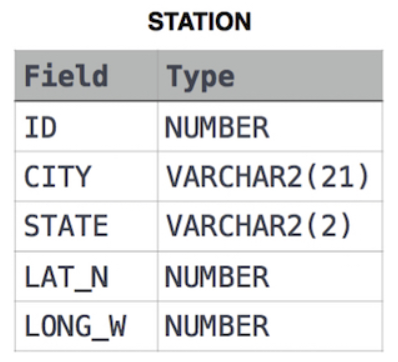
○ LENGTH

[ STATION CITY 가장 긴/ 짧은 길이 출력 문제]

<문제>

STATION 에서 가장 짧은 CITY 이름과 가장 긴 CITY 이름, 그리고 각각의 길이(예: 이름의 문자 수)를 사용 하여 두 도시를 쿼리합니다 . 가장 작은 도시 또는 가장 큰 도시가 두 개 이상인 경우 알파벳순으로 정렬할 때 먼저 오는 도시를 선택하십시오. STATION 테이블은 다음과 같이 설명됩니다.



샘플 입력

예를 들어 CITY 에는 DEF, ABC, PQRS 및 WXY 의 네 가지 항목이 있습니다.

샘플 출력

ABC 3

PQRS 4

설명

알파벳순으로 정렬된 CITY 이름은 길이와 함께 ABC, DEF, PQRS 및 WXY 로 나열됩니다.그리고. 가장 긴 이름은 PQRS 이지만 다음이 있습니다.가장 짧은 이름의 도시에 대한 옵션. ABC 를 선택하십시오 . 알파벳 순서로 처음에 오기 때문입니다.

**SELECT \***

**FROM(**

**SELECT CITY**

**,LENGTH(CITY) AS LEGNTH\_CITY**

**FROM STATION**

**ORDER BY LEGNTH\_CITY, CITY**

**)**

**WHERE ROWNUM = 1**

**;**

**SELECT \***

**FROM(**

**SELECT CITY**

**,LENGTH(CITY) AS LEGNTH\_CITY**

**FROM STATION**

**ORDER BY LEGNTH\_CITY DESC, CITY**

**)**

**WHERE ROWNUM = 1**

**;**

[ Bynary Tree Nodes문제 ]

두 개의 열 N 과 P 를 포함 하는 테이블 BST 가 제공됩니다. 여기서 N 은 Binary Tree 의 노드 값을 나타내고 P 는 N 의 부모입니다 .



노드 값으로 정렬된 Binary Tree 의 노드 유형을 찾는 쿼리를 작성 하십시오. 각 노드에 대해 다음 중 하나를 출력합니다.

• 루트 : 노드가 루트 노드인 경우.

• 리프 : 노드가 리프 노드인 경우.

• 내부 : 노드가 루트 노드도 리프 노드도 아닌 경우.

샘플 입력



샘플 출력

1 Leaf

2 Inner

3 Leaf

5 Root

6 Leaf

8 Inner

9 Leaf

설명

아래 의 이진 트리는 샘플을 보여줍니다.



**SELECT N**

**,CASE WHEN P IS NULL THEN 'Root'**

**WHEN N IN( SELECT DISTINCT P**

**FROM BST) THEN 'Inner'**

**ELSE 'Leaf'**

**ENDd**

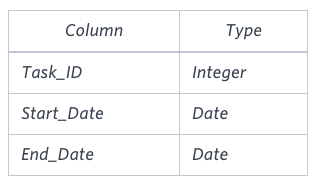
**FROM BST**

**ORDER BY N**

;

[ 날짜 계산 문제 ]

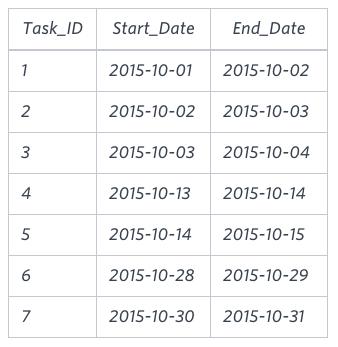
Task\_ID , Start\_Date 및 End\_Date 열이 포함된 Projects 테이블이 제공 됩니다. End\_Date 와 Start\_Date 의 차이 는 테이블의 각 행에 대해 1 일임을 보장합니다.



작업 의 End\_Date 가 연속적이면 동일한 프로젝트의 일부입니다. Samantha는 완료된 서로 다른 프로젝트의 총 수를 찾는 데 관심이 있습니다.

프로젝트를 완료하는 데 걸린 일 수로 나열된 프로젝트의 시작 날짜와 종료 날짜를 오름차순으로 출력하는 쿼리를 작성하십시오. 완료 일수가 동일한 프로젝트가 두 개 이상 있는 경우 프로젝트 시작 날짜를 기준으로 주문하십시오.

샘플 입력



샘플 출력

2015-10-28 2015-10-29

2015-10-30 2015-10-31

2015-10-13 2015-10-15

2015-10-01 2015-10-04

**SELECT S.Start\_Date, E.End\_Date**

**FROM (**

**SELECT Start\_Date**

**,ROW\_NUMBER () OVER(ORDER**

**BY Start\_Date ) AS RNK**

**FROM Projects**

**WHERE Start\_Date NOT IN**

**( SELECT DISTINCT End\_Date**

**FROM Projects**

**)**

**) S**

**,(**

**SELECT End\_Date**

**,ROW\_NUMBER () OVER(ORDER**

**BY End\_Date ) AS RNK**

**FROM Projects**

**WHERE End\_Date NOT IN**

**( SELECT DISTINCT Start\_Date**

**FROM Projects**

**)**

**) E**

**WHERE S.RNK = E.RNK**

**ORDER BY( TO\_DATE(E.End\_Date,'YYYY-MM-DD') - TO\_DATE(S.Start\_Date,'YYYY-MM-DD') )**

**, S.Start\_Date**

**;**

**order by에 두 날짜 사이값 빼는거로 넣어도 ,**

**두 날짜 차이가 적은것 순서대로 정렬됨**

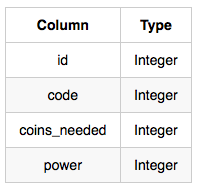
[ 윈도우 함수 문제 ]

헤르미온느는 가장 좋은 선택 방법 은 높은 능력과 나이를 가진 사악하지 않은 지팡이 를 구입하는 데 필요한 최소 골드 갤리온 수를 결정하는 것이라고 결정합니다 .

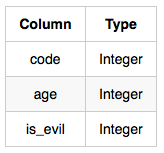
Ron이 관심을 갖고 있는 지팡이의 id , age , coin\_needed 및 power 를 내림차순으로 정렬 하여 출력 하는 쿼리를 작성하십시오 . 2개 이상의 완드가 동일한 위력을 가질 경우 결과를 연령 내림차순으로 정렬합니다 .

<입력 형식>

다음 표에는 올리밴더의 인벤토리에 있는 지팡이에 대한 데이터가 포함되어 있습니다.

-Wands: id 는 완드의 id, code 는 완드 의 코드, coin\_needed 는 완드를 구입하는 데 필요한 골드 갤리온의 총 수이고 power 는 완드의 품질을 나타냅니다(파워가 높을수록 더 좋습니다. 지팡이).

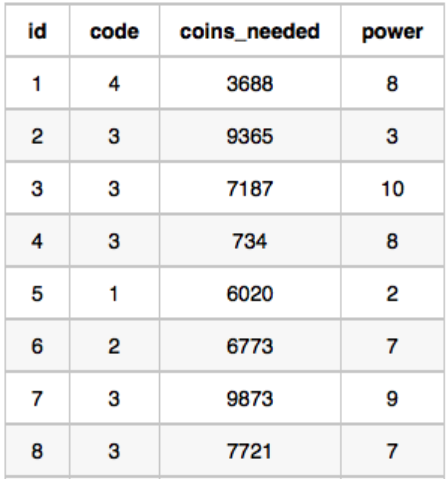
-Wands\_Property: code 는 완드의 코드, age 는 완드의 나이, is\_evil 은 완드가 암흑술에 좋은지 여부를 나타냅니다. is\_evil 의 값 이 0 이면 완드가 사악하지 않다는 의미입니다. 코드 와 연령 간의 매핑 은 일대일입니다.



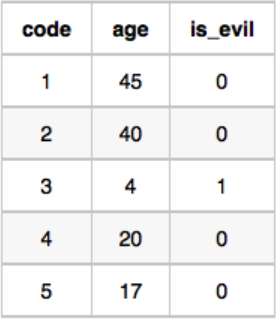
같은 코드 이면 무조건 나이가 같다.

샘플 입력

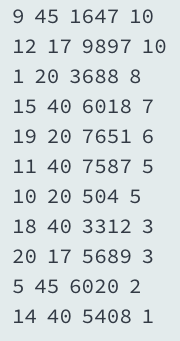
<Wands 테이블>



<Wands\_Property 테이블>



<샘플출력>



**<정답>**

**SELECT K.ID**

**,K.AGE**

**,K.COINS\_NEEDED**

**,K.POWER**

**FROM**

**(**

**SELECT A.ID**

**,B.AGE**

**,A.COINS\_NEEDED**

**,A.POWER**

**,ROW\_NUMBER() OVER(**

**PARTITION BY B.AGE, A.POWER**

**ORDER BY A.COINS\_NEEDED) AS RNK**

**FROM WANDS A**

**,WANDS\_PROPERTY B**

**WHERE A.CODE = B.CODE**

**AND B.IS\_EVIL = '0'**

**) K**

**WHERE K.RNK = 1**

**ORDER BY K.POWER DESC , K.AGE DESC**

**;**

[ RANK 종류 ]

- ROW\_NUMBER() : 1,2,3,4,5,6,7,8,9 ...

- RANK() : 1,1,3,4,4,4,7,7,9....

- DENSE\_RANK() : 1,1,2,3,3,3,4,4,5

점수의 순위를 매기는 SQL 쿼리를 작성하십시오. 순위는 다음 규칙에 따라 계산되어야 합니다.

•점수는 가장 높은 것부터 가장 낮은 것까지 순위가 매겨져야 합니다.

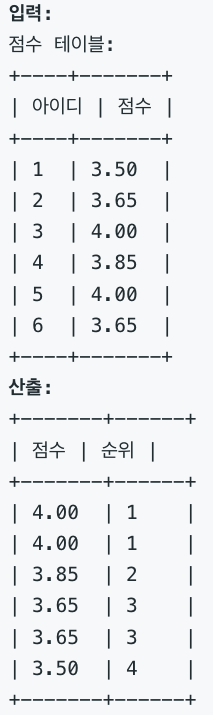
•두 점수 사이에 동점인 경우 둘 다 동일한 순위를 가져야 합니다.

•동점 후 다음 순위 번호는 다음 연속 정수 값이어야 합니다. 즉, 순위 사이에 구멍이 없어야 합니다.

score내림차순 으로 정렬된 결과 테이블을 반환합니다 .

쿼리 결과 형식은 다음 예와 같습니다.

예 1:



**SELECT**

**SCORE**

**,DENSE\_RANK() OVER ( ORDER BY SCORE**

**DESC) AS ‘RANK’**

**FROM SCORES**