Database\_2:

191116

Table name: tb1

|  |  |
| --- | --- |
| name | section |
| abc | CS1 |
| bcd | CS2 |
| abc | CS1 |

1.1. 중복되는 행 찾기

query)

SELECT name, section FROM tb1

GROUP BY name, section

HAVING COUNT(\*) > 1

explanation)

tb1에서 name, section 열로 구성된 행들을 선택한다.

조건절에 COUNT 함수를 사용하기 위해 HAVING을 사용한다.

HAVING을 사용하기 위해 GROUP BY를 사용한다.

중복행을 체크하기 위해 COUNT의 인자로 모든 열에 해당하는 \*를 넘겨준다.

\*SQL 와일드카드: <https://www.w3schools.com/sql/sql_wildcards.asp>

\*SQL GROUP BY 테스트: <https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_groupby_orderby>

\*SQL DISTICT: <https://www.w3schools.com/sql/sql_distinct.asp>

Table name: employees

|  |  |  |
| --- | --- | --- |
| ID | salary | dept |
| 1 | 10000 | EC |
| 2 | 40000 | EC |
| 3 | 30000 | CS |
| 4 | 40000 | ME |
| 5 | 50000 | ME |
| 6 | 60000 | ME |
| 7 | 70000 | CS |

2.1. 두 번째로 큰 salary 구하기

query)

SELECT max(salary) FROM employees WHERE salary IN (SELECT salary FROM employees MINUS SELECT max(salary) FROM employees)

또는

SELECT max(salary) FROM employees WHERE salary <> (SELECT max(salary) FROM employees)

explanation)

첫 번째 쿼리는 최대 임금과의 차이 중 최대값을 찾는다.

두 번째 쿼리는 최대 임금이 아닌 임금 중 최대값을 찾는다. <>: 같지 않다.

\*SQL IN: <https://www.w3schools.com/sql/sql_in.asp>

\*SQL 비교 연산자: <https://www.w3schools.com/sql/sql_operators.asp>

2.2. 다음 쿼리의 결과로 나오는 행의 수는?

query)

SELECT E.ID

FROM employee E

WHERE EXISTS

(SELECT E2.salary FROM employee E2

WHERE E2.DeptName = 'CS'

AND E.salary > E2.salary)

result)

|  |
| --- |
| ID |
| 2 |
| 4 |
| 5 |
| 6 |
| 7 |

의 5행

explanation)

WHERE EXIST는 뒤의 조건이 true인 결과를 반환한다.

문제에서는 자기 자신보다 임금이 낮은 CS 부서 노동자가 있는지가 그 조건이다.

따라서 CS 노동자 최저 임금인 30000 이상을 받는 모든 노동자의 ID가 결과값이 된다.

\*SQL EXIST: <https://www.w3schools.com/sql/sql_exists.asp>

\*SQL 코딩 컨벤션: <https://www.sqlstyle.guide/>

3. dept 테이블에서 업데이트가 일어나는 경우, emp 테이블에서 해당 부서 인원(dept 테이블의 dept\_no를 자신의 dept\_no 값으로 가지고 있는 행)의 sal 값을 100만큼 증가시키는 트리거 작성 (Oracle)

query)

CREATE OR REPLACE TRIGGER my\_trigger

AFTER UPDATE ON dept

FOR EACH ROW

DECLARE

CURSOR emp\_cursor IS SELECT \* FROM emp;

BEGIN

FOR i IN emp\_cursor LOOP

IF i.dept\_no = :NEW.dept\_no THEN

UPDATE emp

SET sal = i.sal + 100

WHERE emp\_no = i.emp\_no;

END IF;

END LOOP;

END;

explanation)

dept 테이블의 행 하나하나가 업데이트 될 때마다 트리거를 실행하기 위해 FOR EACH ROW를 사용한다. (row-level trigger, 행 트리거)

emp 테이블의 행 별 처리를 위해 CURSOR를 설정한다. (이터레이터 개념인듯)

변경된 dept 테이블의 값을 참조하기 위해 ‘:NEW’를 사용한다.

ps) DBMS 별로 커서 문법이 달라서 예제 자체는 크게 의미 없고 이런 기능이 있다를 보시는 게 맞을 듯합니다.

\*Oracle 트리거: <https://www.oracletutorial.com/plsql-tutorial/oracle-trigger/>

\*SQL CURSOR: <https://ko.wikipedia.org/wiki/%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%B2%A0%EC%9D%B4%EC%8A%A4_%EC%BB%A4%EC%84%9C>

\*what is NEW in SQL?: <https://www.reddit.com/r/AskProgramming/comments/87q4o9/what_exactly_is_the_new_keyword_in_sql/>

\*what does ‘:’ do in SQL?: <https://stackoverflow.com/questions/2177978/what-does-the-colon-sign-do-in-a-sql-query>

\*bind variable: <https://www.akadia.com/services/ora_bind_variables.html>

\*SQL Server 트리거: <http://www.sqlservertutorial.net/sql-server-triggers/>

\*MySQL 트리거: <https://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html>

4. student, score 열로 구성된 테이블 t1이 있을 때, 평균 score 보다 큰 score 값을 가진 행 찾기

Query)

SELECT student, score

FROM t1

WHERE score > (SELECT AVG(score) FROM t1)

5. 왜 WHERE 절은 HAVING과 달리 집계함수(aggregate function)와 같이 사용할 수 없는가

WHERE 가 pre-filter 인 반면 HAVING 은 post-filter 이다. 4번 같이 서브쿼리를 사용하지 않는 이상 집계함수보다 먼저 실행된다.

\* <https://www.geeksforgeeks.org/commonly-asked-dbms-interview-questions-set-2/>

6. primary key vs unique key

기본키는 테이블 당 오직 하나이고 null 값을 가질 수 없다. 고유키는 테이블 당 여러 개일 수 있고, null 값을 가질 수 있다.

7. materialized view vs dynamic view

8. embedded SQL vs dynamic SQL

9. CHAR vs VARCHAR