

2. 인덱스 튜닝

- ### 3. 조인 문장 튜닝

- #### 4. 서브쿼리 문장 튜닝

1. 순수하게 서브쿼리로 수행하면서 튜닝
 - [서브쿼리 부더](#) 문제 70 ~ 73
 - [메인쿼리 부더](#) 문제 69 ~ 72
 2. 서브쿼리를 조인으로 변경하게 튜닝
 - 1) in 사용시
 - [nested loop semi join](#) 문제 74
 - [hash sami join](#) 문제 75 ~ 77
 - [merge semi join](#)
 - 2) not in 사용시
 - [nested loop anti join](#)
 - [hash anti join](#) 문제 78 ~ 80
 - [merge anti join](#)
- ※ [qb_name](#) 문제 68

5. 파티션 테이블 생성 및 관리

1. 파티셔닝

- 파티션 뷰 생성 [문제 82 ~ 84](#)
- 파티션 테이블 생성
 - [range 파티션](#) [문제 86](#)
 - [hash 파티션](#) [문제 90](#)
 - [list 파티션](#) [문제 91](#)
 - 복합 파티션

※ 파티션 결과 보는 방법

2. 파티션 pruning [문제 93](#)

3. 파티션 와이즈 조인 [문제 95](#)

- 동적 파티셔닝 [문제 97](#)

4. 인덱스 파티셔닝 [문제 98](#)

- 로컬 파티션 인덱스
바이트, 블록 확인
- 비 파티션 인덱스
- 글로벌 파티션 인덱스

6. 병렬 처리 [문제 99](#)

- 병렬 DML 작업

7. 기타 SQL 튜닝

1. 로지컬 옵티마이저를 제어하는 힌트

- [no_merge, merge](#) [문제 101 ~ 107](#)
- [no_unnest, unnest](#)
- [expand_gset_union](#)
- [no_query_transformation](#)

※ 뷰 쿼리 수정

2. SQL 재작성 [문제 110 ~ 112](#)

=====

※ 테이블 분석 정보 생성일 확인법

※ expand_gset_to_union

※ 실행계획

※ sqlplus 튜닝 툴 팁

※ 테이블 스페이스 오류가 뜬다면 ?

=====

NAME	TYPE	VALUE
cpu_count	integer	8
- 8*2 해서 16까지 줄 수 있다. cpu카운트의 *2 만큼 병렬처리를 줄수 있기 때문 하지만 16을 주면 cpu를 100까지 쓰겠다는 건데 그럼 다른 작업을 할 수 없다 그래서 적당히 높은 숫자를 써주는 것이 좋다 (4 ~ 8)		

2. 인덱스를 생성할 때

예 : create index emp_sal
on emp(sal);

3. 인덱스를 재구성 할때 (테이블이 수정 삭제 되면 인덱스 재구성이 필요 !)

예 : alter index emp_sal rebuild ;

☆인덱스 재구성 테스트

"테이블의 데이터를 수정하게 되면 테이블의 데이터는 변경되지만 인덱스는 변경하지 않고 그냥 남아있게 된다. (ppt 자료 3페이지)

인덱스의 리프블록을 확인하는 방법

analyze index [인덱스 이름] validate structure;

select name, lf_rows, del_lf_rows
from index_stats;

인덱스 재구성 명령어

alter index emp_ename rebuild;

★ index range scan

- index range scan | /*+ index(테이블명 인덱스명) */

★ 결합 컬럼 인덱스

하나의 컬럼으로 인덱스를 구성한게 아니라 여러개의 컬럼으로 인덱스를 구성한 것

※ 현업에서의 인덱스는 대부분 결합컬럼 인덱스가 많다 단일 컬럼 인덱스는 많이 없다 !

예 : create index emp_deptno_sal
on emp(deptno,sal);

```
select deptno, sal, rowid
from emp
where deptno > 0;
```

DEPTNO	SAL	ROWID
10	1300	AAAE+GAABAAALCxAAN
10	2450	AAAE+GAABAAALCxAAC
10	5000	AAAE+GAABAAALCxAAB
20	800	AAAE+GAABAAALCxAAC
20	1100	AAAE+GAABAAALCxAAM
20	2975	AAAE+GAABAAALCxAAD
20	3000	AAAE+GAABAAALCxAAD
20	3000	AAAE+GAABAAALCxAAL
30	950	AAAE+GAABAAALCxAAD
30	1250	AAAE+GAABAAALCxAAD
30	1250	AAAE+GAABAAALCxAAD
30	1500	AAAE+GAABAAALCxAAD
30	1600	AAAE+GAABAAALCxAAD
30	2850	AAAE+GAABAAALCxAAD

Id	Operation	Name
0	SELECT STATEMENT	
* 1	INDEX RANGE SCAN	EMP_DEPTNO_SAL

※ INDEX RANGE SCAN 이 마지막이라는건 인덱스의 모습이라는 뜻이다

※ 이게 결합컬럼 인덱스의 모습이다

※ 생성될 때 정렬의 순서는 왼쪽 컬럼 부터이다.

☆ 결합 컬럼 인덱스의 특징

" 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에 존재해야 결합 컬럼 인덱스를 액세스 할 수 있다."

* scott 유저가 가지고 있는 인덱스 리스트 조회

```
select index_name, column_name, column_position
from user_ind_columns;
```

INDEX_NAME	COLUMN_NAME	COLUMN_POSITION
EMP2_ENAME	ENAME	1
EMP2_TELECOM	TELECOM	1
EMP_DEPTNO_SAL	SAL	2
EMP_DEPTNO_SAL	DEPTNO	1
EMP_JOB_SAL	SAL	2
EMP_JOB_SAL	JOB	1
EMP_SAL	SAL	1
SALES100_EMPNO	EMPNO	1

★ index fast full scan

- index fast full scan | /*+ index_ffs(테이블명 인덱스명) */

※ 그룹함수를 꼭 사용해야 하는 상황이라면 ?

테이블을 full scan 하지말고 index fast full scan 활용

※ index fast full scan 이 되려면 deptno 또는 sal 에 not null 제약이 걸려있어야 한다.

```
alter table emp  
modify deptno not null;
```

※ 만약 not null 제약을 못걸게된 상황이라면 where 절에 null 을 없애는 코드로 짤다

★ index skip scan

- index skip scan | /*+ index_ss(테이블명 인덱스명) */

※ 현업에서의 인덱스는 대부분 결합컬럼 인덱스가 많다 단일 컬럼 인덱스는 많이 없다 !

결합컬럼 인덱스를 사용하려면 인덱스의 첫번째 컬럼이 where 절에 존재해야 결합 컬럼 인덱스를 액세스 할 수 있다

하지만 인덱스의 첫번째 컬럼이 where 절에 없는 상황에서 인덱스를 이용하려면 ?

index skip scan 힌트를 주면 된다 !

★ index unique scan

" unique 한 인덱스를 액세스하는 스캔 방법"

* 인덱스의 종류 2 가지

1. unique 인덱스 : 인덱스를 걸 컬럼의 데이터가 unique 한 데이터인 경우

예 : empno (사원번호)

2. non unique 인덱스 : 인덱스를 걸 컬럼의 데이터가 중복되어 있는 경우의 생성될 인덱스

* unique 인덱스 생성 방법 2 가지

1. 명시적 방법 :

```
create unique index emp_empno  
on emp(empno);
```

2. 암시적 방법 : primary key 와 unique 제약을 걸면 자동으로 unique 인덱스가 생성된다.

* 테스트

1. demobld.sql 스크립트를 수행하시오

2. create unique index emp_empno
on emp(empno);

create index emp_sal
on emp(sal);

3. select index_name, uniqueness
from user_indexes
where table_name = 'EMP';

INDEX_NAME	UNIQUENESS
EMP_EMPNO	UNIQUE
EMP_SAL	NONUNIQUE

4. 다시 demobld 스크립트를 돌린다.

5. insert into emp (empno, ename, sal)
values (7788, 'SCOTT', 3000);

6. create unique index emp_empno
on emp(empno);

ORA-01452: cannot CREATE UNIQUE INDEX; duplicate keys found

※ 설명 : 중복된 데이터가 있으면 unique 인덱스가 안걸린다.

다시 말하면 unique index 가 걸리는 컬럼은 그 컬럼에 중복된 데이터가
하나도 없다는 것이다.

★ index merge scan

- index merge scan | /*+ and_equal(테이블명 인덱스명) */

"데이터를 검색할 때 두개의 인덱스를 동시에 사용해서 더 큰 시너지 효과를 보는 스캔 방법"

* 테스트 :

```
create index emp_job on emp(job);
create index emp_deptno on emp(deptno);

select empno, ename, job, deptno
  from emp
 where job = 'SALESMAN' and deptno = 30;
```

실행계획을 확인하시오 !

```
-----
| Id | Operation                      | Name    |
-----
|  0 | SELECT STATEMENT                |         |
|*  1 |  TABLE ACCESS BY INDEX ROWID   | EMP     |
|*  2 |    INDEX RANGE SCAN              | EMP_JOB |
-----

=====
0  db block gets
4  consistent gets
0  physical reads
=====

              job                      deptno
            COUNT(*)                  COUNT(*)
            -----                  -----
                        4                        6
※ job 이 더 간단하기 때문에 옵티마이저가 job 을 이용했다.
```

실행계획을 보니 job 의 인덱스를 탔는데 만약 두개의 인덱스를 동시에 사용하고 싶다면?

```
select /*+ and_equal(emp emp_deptno emp_job) */ empno, ename, job, deptno
  from emp
 where job = 'SALESMAN' and deptno = 30;
```

```
-----
| Id | Operation                      | Name    |
-----
|  0 | SELECT STATEMENT                |         |
|*  1 |  TABLE ACCESS BY INDEX ROWID   | EMP     |
|  2 |    AND-EQUAL                    |         |
|*  3 |      INDEX RANGE SCAN            | EMP_DEPTNO |
|*  4 |      INDEX RANGE SCAN            | EMP_JOB  |
-----

=====
0  db block gets
8  consistent gets
0  physical reads
=====
```

하지만 요즘에 이것보다 더 강력한 힌트가 있어서 merge scan 은 잘 안쓴다.

★ index bitmap merge scan

```
- index bitmap merge scan | /*+ index_combine(테이블명 인덱스명) */
```

" 두개의 인덱스를 사용하는 것은 index merge scan 과 똑같은데 다른것은 인덱스를 bit 로 변환 해서 사이즈를 확 줄인 다음에 스캔한다는 것이 차이점이다."

예 : 책 앞에 목차가 10 장, 책 뒤의 목차도 10 장 이라고 하면 index merge scan 이 10 장, 10 장 두개를 같이 읽어서 테이블에 찾아갈 rowid 를 알아낸다면
index bitmap merge scan 은 10 장을 1 장으로 요약한다. 1 장. 1 장 두개를 같이 읽어서 테이블에 찾아갈 rowid 를 알아낸다.

```
alter table emp  
  modify job not null;
```

```
alter table emp  
  modify deptno not null;
```

```
select /*+ index_combine(emp emp_job emp_deptno) */ empno, ename, job, deptno  
  from emp  
 where job = 'SALESMAN' and deptno = 30;
```

NESTED LOOPS : 실행계획에 옆과 같이 NESTED LOOPS 가 두번이 나오면 인덱스를 통해서 조인한 결과를 메모리에 올려놓고 다음번에 똑같은 결과를 찾으러 도인할 때 테이블 액세스안하고 메모리에서 찾아서 출력하겠다는 뜻으로 11g 에서 새로 나온 기능인 advanced nested loop join 이라고 한다.

★ hash join

"조인되는 데이터의 양이 대용량일 때 사용하는 조인 방법"
인덱스를 액세스하기보다 full table scan 이 오히려 성능이 유리하다.

☆ 힌트 : use_hash ([탐색 테이블])

※ 해쉬조인은 항상 full 스캔이 따라다녀야 빠르다 !

☆해쉬조인의 원리

```
select /*+leading(d e) use_hash(e) full(d) full(e)*/ e.ename, d.loc
  from emp e, dept d
 where e.deptno = d.deptno;
```

Id	Operation	Name
0	SELECT STATEMENT	
* 1	HASH JOIN	
2	TABLE ACCESS FULL	DEPT
3	TABLE ACCESS FULL	EMP

<← 해쉬 테이블 → 메모리로 올라가는 테이블
<← 탐색 테이블 → 메모리에 올라가지 않는 테이블

둘다 메모리에 올릴 수는 없다.

둘중 하나를 메모리에 올려야 한다면 두개 중 작은 것을
올려야 한다. (조건에 의해서 액세스 되는 건수가 작은것)

테이블이 크면 메모리에 나눠져서 올라가기 때문에 오히려 역효과 !!

그래서 leading 힌트가 굉장히 중요하다 !

앞에 쓴게 해쉬테이블이 된다 !

항상 풀테이블 스캔 해야 검색속도가 높아진다.

왜냐하면 인덱스는 디스크에 있기 때문에 메모리와 디스크를

왔다갔다해서 더 느려진다 !!

해쉬조인을 할때에는 full 을 항상 써야한다. 왜냐하면 나도 모르게

인덱스가 걸려있는 경우도 있고 옵티마이저가 멍청해서 다르게

할 수도 있기 때문이다.

단 ! 탐색테이블에 있는 테이블은 디스크에 있기 때문에 인덱스를

타는게 더 좋을 수 있다.

※ 메모리란 ? 컴퓨터의 메모리 칩

※ 해쉬조인시 유용한 힌트 2 가지 ?

1. swap_join_inputs : 해쉬 테이블을 지정하는 힌트
2. no_swap_join_inputs : 탐색 테이블을 지정하는 힌트

※ 해쉬조인은 조인의 연결고리가 =(이퀄) 조건일 때만 가능하다.

만약에 emp 와 salgrade 가 대용량 테이블이어서 조인 성능이 느리다면 반드시 해쉬조인을
사용해야 하는데 사용 못하는 상황이면 아래와 같이 sort merge join 을 수행해야한다.

★ sort merge join

힌트 : use_merge([탐색 테이블])

☆ sort merge join 의 원리

"연결고리가 되는 컬럼의 데이터를 정렬해서 조인하는 조인 방법"
대용량 데이터를 조인할 때 유리한 조인 방법

```
select e.ename, d.loc, e.deptno
  from emp e, dept d
 where e.deptno = d.deptno;
    10 ↓ <-----10
    10 ↓
    10 ↓ 여기까지

    20 ↓ <-----20
    20 ↓
    20 ↓
    20 ↓ 여기까지

    30 ↓ <-----30
    30 ↓
    30 ↓
    30 ↓ 여기까지
      :
      : ↓ <-----40
```

정렬을 해놓기 때문에 모든 테이블을 찾지 않는다 !

★ outer join 튜닝

※ outer join 의 조인 순서는 항상 outer join 사인이 없는 쪽에서 있는쪽으로 조인한다.
그래서 swap_join_inputs([해쉬 테이블]) 힌트를 써서 조인순서를 변경할 수 있다.

★ full outer join 튜닝

1. 예측 실행 계획 : SQL을 실행해보기 전에 미리 예측한 실행계획
2. 실제 실행 계획 : SQL을 실행하고 실행할 때 사용했던 실행계획

※ qb_name 힌트에 대한 결과는 실제 실행계획에서만 볼 수 있다.

★★★ push_subq 나 no_push_subq 힌트를 사용하려면 no_unnest 와 같이 사용해야 한다.

※ (=)조건절이라면 별다른 힌트를 안줘도 잘 수행한다 왜냐하면 (=)을 써서 1개만 리턴하기 때문이다.
하지만 in 을 쓰게 된다면 여러개를 리턴하기 때문에 대용량이라 판단하고
옵티마이저가 조인문으로 바꿔버린다.

서브쿼리로 수행되는 SQL 이 양쪽 다 대용량이면 서브쿼리로 수행되는것 보다는 조인으로 수행되게
하는 것이 더 성능이 좋다. 조인 방법중에 해쉬조인을 사용할 수 있기 때문이다.

※ 서브쿼리 메인쿼리 둘중 한쪽이 대용량인경우 소용량인 쿼리부터 풀게 서브쿼리문을 사용하지만
둘다 대용량이라면 조인으로 푸는 것이 더 현명하다

※ 웬만하면 서브쿼리쪽에 힌트를 작성하는데 만약 원하는 결과가 나오지 않는다면
메인쿼리에도 써봐라!

예 : 우리반 테이블을 통신사 별로 파티셔닝 하고 싶다.
sk, kt, lg

4. 복합 파티션

range - hash 파티션
range - list 파티션
range - range 파티션
list - hash 파티션
list - list 파티션
list - range 파티션

※ 중요한 점! 파티션 테이블의 효과를 보려면 파티션마다 데이터가 골고루 잘 분포가 되어 있어야 한다.

☆ 파티션 뷰 생성 -----> [문제 82 ~ 84](#)

☆ 파티션 결과 보는 방법

```
exec dbms_stats.gather_table_stats('SCOTT', 'EMP195');
```

```
select table_name, partition_name, num_rows  
from user_tab_partitions  
where table_name = 'EMP195';
```

TABLE_NAME	PARTITION_NAME	NUM_ROWS
EMP195	SYS_P286	6
EMP195	SYS_P287	3
EMP195	SYS_P288	5

★ 파티션 pruning

prune 의 뜻 ? 쓸데 없는 가지를 치다
불필요한 부분을 제거한다.

SQL 을 실행하는 시점에서 SQL 의 조건절을 분석해서 읽지 않아도 되는 파티션 세그먼트를 액세스 대상에서 제외시키는 기능

* 파티션 프루닝이 되고 있는 실행계획

```
select *
  from emp_partition2
 where deptno = 20;
```

경 과: 00:00:00.01

Id	Operation	Name
0	SELECT STATEMENT	
1	PARTITION RANGE SINGLE	
* 2	TABLE ACCESS FULL	EMP_PARTITION2

0 db block gets
4 consistent gets
0 physical reads

※ PARTITION RANGE SINGLE -----> 파티션 프루닝이 되고 있다.
즉 옷장의 해당 서랍만 열었다.

☆ 파티션 프루닝이 안되는 경우

1. where 절의 조건 컬럼을 가공했을 때

```
select /*+ gather_plan_statistics */ *
  from emp_partition2
 where trim(deptno) = 20;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		5	00:00:00.01	10
1	PARTITION RANGE ALL		1	1	5	00:00:00.01	10
* 2	TABLE ACCESS FULL	EMP_PARTITION2	3	1	5	00:00:00.01	10

2. 등차조건이나 in 조건이 아닐 때 (hash 파티션일때만)

```
select /*+ gather_plan_statistics */ *
  from emp195
 where deptno like '2%';
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		5	00:00:00.01	10
1	PARTITION HASH ALL		1	1	5	00:00:00.01	10
* 2	TABLE ACCESS FULL	EMP195	3	1	5	00:00:00.01	10

★ 파티션 와이즈 조인 (wise join)

"파티션끼리 조인 하는 것"

	10	20	30	40	
emp_partition	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	↓	↓	↓	↓	----> 파티션 끼리 조인
dept_partition	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

* 파티션 와이즈 조인이 아닌 경우

emp 10 20 10 20 30 10 20 20

↑

dept 10 20 30 40

※ 해쉬조인으로 유도를 해야 파티션 와이즈조인을 한다.

조인하려는 테이블이 둘다 파티션 테이블인데 파티션 와이즈 조인을 안한다면 `pq_distribute(e, none, none)`를 사용해야 한다. (e 는 해쉬의 디스크 테이블)

☆ 둘다 파티션 테이블이 아닌 경우 (동적 파티셔닝)

조인 : outer table : inner table
 (dept) (emp)
 파티션 x 파티션 x
 big table big table

힌트 : `pq_distribute (emp, hash, hash)`

테이블을 둘다 파티셔닝 하고 full partition wise 조인 해라 ~

그럼 파티션 테이블을 안만들면 되겠네~

답 ! 이것은 파티션 만들 때 메모리를 많이 써서 좋은 방법은 아니다.

```
select /*+ leading(d e) use_hash(e) full(d) full(e)
       parallel(e 2) parallel(d 2)
       pq_distribute(e, hash, hash) */
       e.ename, d.loc, e.deptno
from emp e, dept d
where e.deptno = d.deptno;
```

※ 연결고리를 기준으로 파티션을 나눈다. 그리고 대용량이 되면 그냥 조인하는것 보다는 확실히 빠르기 때문에 파티션 와이즈 조인을 쓰는 것이다!

※ 파티셔닝을 해주는 힌트를 쓰려면 병렬 힌트도 꼭 같이 써야한다.

왜냐하면 병렬처리 때문에 나누는 부분에 파티션한 조각들이 들어가기 때문이다.

그래서 병렬을 같이 붙이는데 병렬 개수는 파티션을 할 수 있게 짝수로만 써주는게 좋다.

★ 인덱스 파티셔닝

"인덱스의 크기가 너무 커서 인덱스를 파티션해서 성능을 높이겠다."

* 인덱스 파티션의 종류

1. 로컬 파티션 인덱스




"파티션 테이블의 파티션에 각각 로컬 인덱스로 구성된 인덱스"

2. 비 파티션 인덱스

"테이블은 파티션 되어져 있는데 인덱스는 파티셔닝 되지 않은 인덱스"

3. 글로벌 파티션 인덱스

"테이블도 파티션 되어져 있고 인덱스도 파티션 되어있는데 테이블 파티션과 인덱스 파티션과의 관계가 서로 독립적 구조로 되어있는 인덱스 "

구분	테이블	인덱스		
		비파티션	파티션	
			글로벌	로컬
비파티션 테이블				
파티션 테이블				

※ 각자 장단점이 있다. 로컬이 가장 쓰기 편해보이지만 단점은 저장공간을 많이 차지한다.

* 로컬 파티션 인덱스

1. prefixed 파티션 인덱스 : deptno + job

↑
파티션 키 컬럼

```
drop index emp_partition_index;
```

```
create index emp_partition_index
on emp_partition(deptno, job) local;
```

※ 파티션 키 컬럼이 선두에 있는 결합 컬럼 인덱스가 prefixed

2. non prefixed 파티션 인덱스 : job + deptno

↑
파티션 키 컬럼

```
drop index emp_partition_index
```

```
create index emp_partition_index
on emp_partition(job, deptno) local;
```

테이블은 deptno 를 기준으로 파티션 되어있는데 SQL 에 where 절에 주로 job 을 검색한다고 하면 non partition index 가 필요하다.

* 비 파티션 인덱스

```
drop index emp_partition_local;
```

```
create index emp_partition_index  
on emp_partition(deptno) ;
```

※ 로컬 파티션 인덱스와 비 파티션 인덱스의 장단점

- 로컬 파티션 인덱스 : 장점 - 관리가 쉽다. 검색 성능이 좋다
단점 - 공간을 많이 사용한다.
- 비 파티션 인덱스 : 장점 - 공간을 적게 사용한다.
단점 - 관리가 불편하다.

* 어떤점이 불편한가?

비 파티션 인덱스에 관련된 파티션 테이블을 drop 했을 때 비 파티션 인덱스가 invalid 되어서 인덱스를 rebuild 해줘야 한다.

```
alter table emp_partition  
drop partition p1;
```

```
select index_name, status  
from user_indexes  
where index_name = 'EMP_PARTITION_INDEX';
```

INDEX_NAME	STATUS
EMP_PARTITION_INDEX	UNUSABLE

※ 이렇게 되어버리면 이 인덱스는 못쓴다.

```
select /*+ index(e emp_partition_index) */ ename, sal  
from emp_partition e  
where deptno = 20;
```

ORA-01502: 인덱스 'SCOTT.EMP_PARTITION_INDEX' 또는
인덱스 분할영역은 사용할 수 없는 상태입니다

그래서 rebuild 를 해줘야 한다.

※대용량이라면 엄청 오래 걸린다.

```
alter index emp_partition_index rebuild;
```

```
select /*+ index(e emp_partition_index) */ ename, sal  
from emp_partition e  
where deptno = 20;
```

ENAME	SAL
JONES	2975
FORD	3000
SMITH	800

```
create table emp302
as
select *                               /* 테이블 구조 생성 */
  from emp
 where 1=2;

insert into emp302                     /* 테이블 값 입력 */
select * from emp;
```

```
select * from emp302;
```

```
delete from emp302;      /* 등성등성 지워야 하는데 그냥 전부 지웠다. */  
                        /* high water mark 의 높이는 그대로 */  
commit;
```

※ high water mark 는 truncate 를 해야 높이가 내려간다.

* high water mark 위로 데이터를 입력하는 방법

1. append 힌트 : high water mark 위로 데이터를 입력하겠다.

```
insert /*+ append */ into emp302  
select * from emp;
```

```
select * from emp302;
```

ORA-12838: 병렬로 수정한 후 객체를 읽거나 수정할 수 없습니다 r

※ high water mark 위로 데이터를 넣었기 때문에 commit 을 해야
high water mark 의 높이가 넣은데이터 위로 올라가서 검색이 가능하다.

☆ 저장공간 낭비가 생기지만 빨리 insert 할 수 있다.

2. parallel 힌트 : high water mark 위로 데이터를 입력하는데 병렬로 입력하겠다.

```
delete from emp302;  
commit;
```

```
alter session enable parallel dml;      /* 이 명령어를 날려줘야 병렬 insert 가능  
                                         */
```

```
insert /*+ parallel(e3 4) */ into emp302 e3  
select * from emp;
```

```
commit;
```

```
select * from emp302;
```

분석함수를 이용하지 않은 SQL ----> 분석함수를 이용한 SQL
 <----

※ 읽는 순서가 안쪽에서부터 읽는다

☆ 실행계획 보는 방법 2 가지

1. 예상 실행계획 보는 방법 ---> SQL gate 에서 F7 누르면 된다.
2. 실제 실행계획 보는 방법 ---> 실제로 실행을 하면서 실행할때 사용한 수행계획 확인

ed p.sql

```
select *  
from table(dbms_xplan.display_cursor(null,null,'ALLSTATS LAST +alias +outline +predicate'));
```

```
select /*+ gather_plan_statistics */ *  
  from emp_partition2  
  where deptno = 20;
```

@p

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		5	00:00:00.01	4
1	PARTITION RANGE SINGLE		1	5	5	00:00:00.01	4
* 2	TABLE ACCESS FULL	EMP_PARTITION2	1	5	5	00:00:00.01	4

※ 버퍼가 우리가 항상 보던 블록 수를 의미한다 그래서 이것을 보고 버퍼가 크면 병목부분이라고 하고 이곳을 중심으로 튜닝을 진행한다.

※ sql plus 튜닝 툴 소환

set timing on : 전체 걸린시간을 출력

☆☆☆테이블 스페이스 오류가 뜬다면

ORA-01652: unable to extend temp segment by 128 in tablespace SYSTEM

```
alter tablespace system  
add datafile 'c:\system02.dbf' size 300m;
```

Id	Operation	Name
0	SELECT STATEMENT	
1	TABLE ACCESS BY INDEX ROWID	EMP
* 2	INDEX RANGE SCAN	EMP_ENAME

문제 4. 이름이 SCOTT인 사원의 이름과 월급과 직업을 출력하는 SQL을 작성하는데 full table scan이 되게 힌트를 주시오 !

```
select /*+ full(emp) */ ename, sal, job
  from emp
 where ename = 'SCOTT';
```

Id	Operation	Name
0	SELECT STATEMENT	
* 1	TABLE ACCESS FULL	EMP

문제 5. 위의 SQL의 full table scan 실행계획과 인덱스 스캔 실행계획의 블록의 차이가 얼마나 발생하는지 확인하시오 !

튜닝전 :

```
select /*+ full(emp) */ ename, sal, job
  from emp
 where ename = 'SCOTT';
```

```
=====
0  db block gets
4  consistent gets
0  physical reads
=====
```

튜닝후 :

```
select /*+ index(emp emp_ename) */ ename, sal, job
  from emp
 where ename = 'SCOTT';
```

```
=====
0  db block gets
3  consistent gets
0  physical reads
=====
```

문제 6. 오라클 교육용 데이터중에 가장 큰 sales 테이블을 가지고 아래의 테이블을 생성하시오 !

```
create table sales100
as
select rownum empno, s.*
  from sales s;
```

EMPNO	PROD_ID	CUST_ID	TIME_ID	CHANNEL_ID	PROMO_ID	QUANTITY_SOLD	AMOUNT_SOLD
1	13	987	1998-01-10	3	999	1	1232.16
2	13	1660	1998-01-10	3	999	1	1232.16
3	13	1762	1998-01-10	3	999	1	1232.16
4	13	1843	1998-01-10	3	999	1	1232.16
5	13	1948	1998-01-10	3	999	1	1232.16

문제 7. sales100의 직원번호가 56071번인 사원의 모든 컬럼을 출력하는 쿼리를 작성하고 읽어들이는 블록의 갯수를 확인하시오 !

```
select *
  from sales100
 where empno = 56071;
```

```
=====
EMPNO      PROD_ID    CUST_ID TIME_ID  CHANNEL_ID  PROMO_ID QUANTITY_SOLD  AMOUNT_SOLD
-----
56071         38        1777 98/06/15         3         999             1         34.06
=====
```

```
=====
      0  db block gets
5065  consistent gets
5060  physical reads
=====
```

문제 8. sales100의 직원번호에 인덱스를 걸고 위의 SQL을 실행해서 인덱스 스캔을 하는지 확인을 하고 블록의 갯수를 확인하시오 !

```
create index sales100_empno
on sales100(empno);
```

```
select *
  from sales100
 where empno = 56071;
```

```
=====
      0  db block gets
      5  consistent gets
      0  physical reads
=====
```

문제9. KING의 이름을 JACK으로 변경하고 commit하시오 !

```
update emp
set ename = 'JACK'
where ename = 'KING';
```

```
commit;
```

테이블의 경우는 데이터가 KING ---> JACK 으로 바로 변경이 되나
인덱스는 KING 리프블록 그냥 두고 JACK이라는 리프블록을 추가한다.

```
analyze index emp_ename validate structure;
```

```
select name, lf_rows, del_lf_rows
  from index_stats;
```

```
=====
NAME                                                    LF_ROWS DEL_LF_ROWS
-----
EMP_ENAME                                                    15         1
=====
```

※ 테이블의 데이터 수는 14개 (인덱스에서는 하나 추가되고 하나 죽었음)

☆ 인덱스 재구성 명령어

```
alter index emp_ename rebuild;
analyze index emp_ename validate structure;
```

```
select name, lf_rows, del_lf_rows
from index_stats;
```

NAME	LF_ROWS	DEL_LF_ROWS
EMP_ENAME	14	0

- ※ 인덱스가 정리 되었음 !
- ※ 주기적으로 정리를 해줘야 함 !

문제 10. 월급이 3000인 사원의 이름과 월급을 출력하는 SQL을 작성하는데 실행계획이 index range scan이 되도록 인덱스를 걸고 작성하시오 !

```
select /*+ index(emp emp_sal) */ ename, sal
from emp
where sal = 3000;
```

ENAME	SAL
FORD	3000
SCOTT	3000

Id	Operation	Name
0	SELECT STATEMENT	
1	TABLE ACCESS BY INDEX ROWID	EMP
* 2	INDEX RANGE SCAN	EMP_SAL

- ※ 월급이 3000인것들의 범위를 스캔 해서 rowid(페이지번호) 를 통해 테이블에서 원하는 자료를 찾는것

인덱스	테이블
800 AAAE+GAABAAALCxAAC	[AAAE+GAABAAALCxAAC JACK] 검색
950 AAAE+GAABAAALCxAAH	AAAE+GAABAAALCxAAB BLAKE
1100 AAAE+GAABAAALCxAAM	AAAE+GAABAAALCxAAC CLARK
1250 AAAE+GAABAAALCxAAC	AAAE+GAABAAALCxAAD JONES
1250 AAAE+GAABAAALCxAAI	AAAE+GAABAAALCxAAC MARTIN
1300 AAAE+GAABAAALCxAAN	AAAE+GAABAAALCxAAC ALLEN
1500 AAAE+GAABAAALCxAAG	AAAE+GAABAAALCxAAG TURNER
1600 AAAE+GAABAAALCxAAC	AAAE+GAABAAALCxAAC JAMES
2450 AAAE+GAABAAALCxAAC	AAAE+GAABAAALCxAAC WARD
2850 AAAE+GAABAAALCxAAB	-----> [AAAE+GAABAAALCxAAC FORD] 검색
2975 AAAE+GAABAAALCxAAD	↗ AAAE+GAABAAALCxAAC SMITH
[3000 AAAE+GAABAAALCxAAC]----- ↗	↗ [AAAE+GAABAAALCxAAC SCOTT] 검색
[3000 AAAE+GAABAAALCxAAC]-----	AAAE+GAABAAALCxAAC ADAMS
[5000 AAAE+GAABAAALCxAAC]	AAAE+GAABAAALCxAAC MILLER

스캔

문제 11. 우리반 테이블에 이름에 인덱스를 걸고 아래의 SQL이 인덱스를 통해서 어떻게 테이블의 데이터를 액세스 하는지 그림으로 그리시오 !

```
create index emp2_ename on emp2(ename);
```

```
select ename, age, major
from emp2
where ename = '서일';
```

ENAME	AGE	MAJOR
서일	26	심리학과

emp2_ename 인덱스 ----- emp2 테이블

```
select ename, rowid
from emp2
where ename > ' ';
```

ENAME	ROWID		ROWID	ENAME
김건희	AAAE6LAABAAALDJAEE		AAAE6LAABAAALDJAAB	엄한솔
김용식	AAAE6LAABAAALDJAAY		AAAE6LAABAAALDJAAC	김준구
김용원	AAAE6LAABAAALDJAHH	↑	AAAE6LAABAAALDJAAD	김준하
김준구	AAAE6LAABAAALDJAAC	↑	AAAE6LAABAAALDJAEE	김건희
김준하	AAAE6LAABAAALDJAAD	↑	AAAE6LAABAAALDJAFF	이후림
김진	AAAE6LAABAAALDJAAb	↑	AAAE6LAABAAALDJAAG	박태균
김진철	AAAE6LAABAAALDJAAW	↑	AAAE6LAABAAALDJAHH	김용원
김혜진	AAAE6LAABAAALDJAAM	↑	AAAE6LAABAAALDJAII	장보겸
박태균	AAAE6LAABAAALDJAAG	↗	AAAE6LAABAAALDJAII	장보겸
[서일	AAAE6LAABAAALDJAAB]-----		AAAE6LAABAAALDJAII	장보겸
[신선혜	AAAE6LAABAAALDJAAB]		AAAE6LAABAAALDJAII	장보겸
안우용	AAAE6LAABAAALDJAAX		AAAE6LAABAAALDJAII	장보겸
안혜진	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸
엄한솔	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸
오세희	AAAE6LAABAAALDJAAS		AAAE6LAABAAALDJAII	장보겸
유이수	AAAE6LAABAAALDJAAP		AAAE6LAABAAALDJAII	장보겸
이상엽	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸
이서영	AAAE6LAABAAALDJAAR		AAAE6LAABAAALDJAII	장보겸
이소진	AAAE6LAABAAALDJAAN		AAAE6LAABAAALDJAII	장보겸
이후림	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸
임혜진	AAAE6LAABAAALDJAAL		AAAE6LAABAAALDJAII	장보겸
장보겸	AAAE6LAABAAALDJAII		AAAE6LAABAAALDJAII	장보겸
정성호	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸
정지엽	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸
주소현	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸
최원형	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸
최재혁	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸
허석우	AAAE6LAABAAALDJAAB		AAAE6LAABAAALDJAII	장보겸

문제 12. 통신사 컬럼에 인덱스를 생성하시오 !

```
create index emp2_telecom
on emp2(telecom);
```

문제 13. 이름이 서일이고 통신사가 sk인 학생의 이름과 통신사와 나이와 전공을 출력하시오 !

```
select ename, telecom, age, major
from emp2
where ename = '서일' and telecom = 'sk';
```

```
=====
ENAME    TELECOM                AGE MAJOR
-----
서일     sk                    26 심리학과
=====
```

```
-----
| Id | Operation                | Name          |
-----
|  0 | SELECT STATEMENT          |               |
|*  1 | TABLE ACCESS BY INDEX ROWID| EMP2          |
|*  2 | INDEX RANGE SCAN          | EMP2_TELECOM  |
-----
```

※ 똑똑한 옵티마이저이면 ename을 타는게 더 빠르다는걸 알고 간다

문제 14. 아래의 SQL의 인덱스가 한번은 ename의 인덱스를 타게 힌트를 주고 또 한번은 telecom의 인덱스를 타게끔 힌트를 주시오 !

답1

```
select /*+ index(emp2 emp2_ename) */ ename, telecom, age, major
from emp2
where ename = '서일' and telecom = 'sk';
```

```
-----
| Id | Operation                | Name          |
-----
|  0 | SELECT STATEMENT          |               |
|*  1 | TABLE ACCESS BY INDEX ROWID| EMP2          |
|*  2 | INDEX RANGE SCAN          | EMP2_ENAME    |
-----
```

답2

```
select /*+ index(emp2 emp2_telecom) */ ename, telecom, age, major
from emp2
where ename = '서일' and telecom = 'sk';
```

```
-----
| Id | Operation                | Name          |
-----
|  0 | SELECT STATEMENT          |               |
|*  1 | TABLE ACCESS BY INDEX ROWID| EMP2          |
|*  2 | INDEX RANGE SCAN          | EMP2_TELECOM  |
-----
```

문제 15. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
select ename, telecom, age
  from emp2
 where lower(telecom) = 'sk';
```

```
-----
| Id | Operation          | Name |
-----
|  0 | SELECT STATEMENT   |      |
|*  1 |  TABLE ACCESS FULL| EMP2  |
-----
```

```
=====
0  db block gets
4  consistent gets
0  physical reads
=====
```

튜닝후 :

```
select ename, telecom, age
  from emp2
 where telecom in ('sk', 'SK', 'sK', 'Sk' );
```

```
-----
| Id | Operation                                | Name          |
-----
|  0 | SELECT STATEMENT                        |               |
|  1 |  INLIST ITERATOR                       |               |
|  2 |    TABLE ACCESS BY INDEX ROWID        | EMP2          |
|*  3 |      INDEX RANGE SCAN                   | EMP2_TELECOM  |
-----
```

```
=====
0  db block gets
5  consistent gets
0  physical reads
=====
```


문제 16. 아래의 sql을 튜닝하시오 !

튜닝전 :

```
select ename, age, telecom
  from emp2
 where telecom || age = 'sk26';
```

```
-----
| Id | Operation          | Name |
-----
|  0 | SELECT STATEMENT    |      |
|*  1 |  TABLE ACCESS FULL| EMP2  |
-----
```

```
=====
0  db block gets
4  consistent gets
0  physical reads
=====
```

튜닝후 :

```
select ename, age, telecom
  from emp2
 where telecom = 'sk' and age = 26;
```

```
-----
| Id | Operation                                | Name          |
-----
|  0 | SELECT STATEMENT                        |               |
|*  1 |  TABLE ACCESS BY INDEX ROWID          | EMP2          |
|*  2 |    INDEX RANGE SCAN                     | EMP2_TELECOM  |
-----
```

```
=====
0  db block gets
4  consistent gets
0  physical reads
=====
```

문제 17. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
select ename, sal, hiredate
  from emp
 where to_char ( hiredate, 'rr/mm/dd' ) = '81/11/17';
```

```
=====
ENAME                               SAL HIREDATE
-----
JACK                               5000 81/11/17
=====
```

```
-----
| Id | Operation                | Name |
-----
|  0 | SELECT STATEMENT         |      |
|*  1 |  TABLE ACCESS FULL      | EMP  |
-----
```

```
=====
0  db block gets
4  consistent gets
0  physical reads
=====
```

튜닝 후 :

```
create index emp_hiredate
  on emp(hiredate);

select ename, sal, hiredate
  from emp
 where hiredate = to_date( '81/11/17', 'rr/mm/dd' );
```

```
=====
ENAME                               SAL HIREDATE
-----
JACK                               5000 81/11/17
=====
```

```
-----
| Id | Operation                | Name |
-----
|  0 | SELECT STATEMENT         |      |
|  1 |  TABLE ACCESS BY INDEX ROWID | EMP  |
|*  2 |    INDEX RANGE SCAN       | EMP_HIREDATE |
-----
```

```
=====
0  db block gets
3  consistent gets
0  physical reads
=====
```

문제 18. 아래의 SQL을 튜닝하시오 !

(그룹함수 쓰지 말고 인덱스로만 원하는 결과를 볼 수 있게 하시오 !)

튜닝전 :

```
select deptno, max(sal)
  from emp
 where deptno = 10
 group by deptno;
```

```
=====
DEPTNO    MAX(SAL)
-----
      10      5000
=====
```

```
-----
| Id | Operation                | Name                |
-----
|  0 | SELECT STATEMENT          |                     |
|  1 |   SORT GROUP BY NOSORT    |                     |
|*  2 |    INDEX RANGE SCAN       | EMP_DEPTNO_SAL     |
-----
```

```
=====
0  db block gets
1  consistent gets
0  physical reads
=====
```

튜닝 후 :

```
select /*+ index_desc(emp emp_deptno_sal) */ deptno, sal
  from emp
 where deptno = 10
    and rownum = 1;
```

```
=====
DEPTNO      SAL
-----
      10      5000
=====
```

```
-----
| Id | Operation                                | Name                |
-----
|  0 | SELECT STATEMENT                        |                     |
|*  1 |   COUNT STOPKEY                        |                     |
|*  2 |    INDEX RANGE SCAN DESCENDING         | EMP_DEPTNO_SAL     |
-----
```

```
=====
0  db block gets
1  consistent gets
0  physical reads
=====
```

문제 19. 아래의 SQL을 튜닝하시오 !
 (필요한 인덱스도 알아서 생성하시오 !)

튜닝전 :

```
select job, min(sal)
  from emp
 where job = 'SALESMAN'
 group by job;
```

JOB	MIN(SAL)
SALESMAN	1250

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT GROUP BY NOSORT	
* 2	INDEX RANGE SCAN	EMP_JOB_SAL


```
0 db block gets
1 consistent gets
0 physical reads
```

튜닝후 :

```
create index emp_job_sal
  on emp(job, sal);

select /*+ index(emp emp_job_sal) */ job, sal
  from emp
 where job = 'SALESMAN'
    and rownum = 1;
```

JOB	SAL
SALESMAN	1250

Id	Operation	Name
0	SELECT STATEMENT	
* 1	COUNT STOPKEY	
* 2	INDEX RANGE SCAN	EMP_JOB_SAL


```
0 db block gets
1 consistent gets
0 physical reads
```

※이게 왜 튜닝이냐 !

그룹함수를 쓴다는거는 일단 테이블을 모두 봐야하는 것이다.

문제 20. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
select deptno, sum(sal)
  from emp
 group by deptno;
```

```
=====
   DEPTNO    SUM(SAL)
-----
        30         9400
        20        10875
        10         8750
=====
```

```
-----
| Id | Operation                | Name |
-----
|  0 | SELECT STATEMENT          |      |
|  1 |  HASH GROUP BY            |      |
|  2 |    TABLE ACCESS FULL EMP |      |
-----
```

```
=====
0  db block gets
3  consistent gets
0  physical reads
=====
```

튜닝후 :

```
select /*+ index_ffs(emp emp_deptno_sal) */ deptno, sum(sal)
  from emp
 where deptno >=0
 group by deptno;
```

```
=====
   DEPTNO    SUM(SAL)
-----
        30         9400
        20        10875
        10         8750
=====
```

```
-----
| Id | Operation                | Name |
-----
|  0 | SELECT STATEMENT          |      |
|  1 |  HASH GROUP BY            |      |
|*  2 |    INDEX FAST FULL SCAN EMP_DEPTNO_SAL |      |
-----
```

```
=====
0  db block gets
3  consistent gets
0  physical reads
=====
```

※ 그룹함수를 꼭 사용해야 하는 상황이라면 ?

테이블을 full scan 하지말고 index fast full scan 활용

※ index fast full scan이 되려면 deptno 또는 sal 에 not null 제약이 걸려있어야 한다.

```
alter table emp
  modify deptno not null;
```

※ 만약 not null 제약을 못걸게된 상황이라면 where 절에 null을 없애는 코드로 짤다

문제 21. 아래의 SQL을 튜닝하시오 !
 (인덱스와 힌트를 알아서 생성하시오 !)

튜닝전 :

```
select job, count(*)
  from emp
 group by job;
```

JOB	COUNT(*)
SALESMAN	4
CLERK	4
PRESIDENT	1
MANAGER	3
ANALYST	2

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
2	TABLE ACCESS FULL	EMP

```
0 db block gets
3 consistent gets
0 physical reads
```

튜닝후 :

```
alter table emp
  modify job not null;
```

```
select /*+ index_ffs(emp emp_job_sal) */ job, count(*)
  from emp
 group by job;
```

JOB	COUNT(*)
CLERK	4
SALESMAN	4
PRESIDENT	1
MANAGER	3
ANALYST	2

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
2	INDEX FAST FULL SCAN	EMP_JOB_SAL

```
0 db block gets
3 consistent gets
0 physical reads
```

=====

* 다시 demobld 스크립트 돌리세요 !

```
create index emp_deptno_sal
on emp(deptno, sal);
```

```
select index_name, column_name, column_position
from user_ind_columns
where table_name = 'EMP'
order by index_name, column_position;
```

INDEX_NAME	COLUMN_NAME	COLUMN_POSITION
EMP_DEPTNO_SAL	DEPTNO	1
EMP_DEPTNO_SAL	SAL	2

=====

문제 22. 월급이 1250인 사원의 이름과 월급과 부서번호를 출력하고 실행계획을 확인하시오!

```
select ename, sal, deptno
from emp
where sal = 1250;
```

ENAME	SAL	DEPTNO
MARTIN	1250	30
WARD	1250	30

=====

Id	Operation	Name
0	SELECT STATEMENT	
* 1	TABLE ACCESS FULL	EMP

=====

0 db block gets
4 consistent gets
0 physical reads

=====

※ where 절에 결합 컬럼 인덱스의 첫번째 컬럼이 없어서 TABLE ACCESS FULL 했다.

※ 현업에서의 인덱스는 대부분 결합컬럼 인덱스가 많다 단일 컬럼 인덱스는 많이 없다 !

```
select /*+ index_ss(emp emp_deptno_sal) */ ename, sal, deptno
from emp
where sal = 1250;
```

ENAME	SAL	DEPTNO
MARTIN	1250	30
WARD	1250	30

=====

Id	Operation	Name
0	SELECT STATEMENT	

=====

1	TABLE ACCESS BY INDEX ROWID	EMP
* 2	INDEX SKIP SCAN	EMP_DEPTNO_SAL

```
=====
0  db block gets
4  consistent gets
0  physical reads
=====
```

※ 설명 : 스킵스캔은 인덱스를 처음부터 스캔하는데 순서가
 1컬럼(10)에서 2컬럼 1250값을 스캔 ----> 없음
 1컬럼(20)에서 2컬럼 1250값을 스캔 ----> 없음
 1컬럼(30)에서 2컬럼 1250값을 스캔 ----> 1250 찾음
 ---->1컬럼(30)에서는 스캔 끝! 건너뛰기!

참고 !	DEPTNO	SAL
	10	1300 <--스캔
	10	2450 <--스캔
	10	5000 <--스캔
	20	800 <--스캔
	20	1100 <--스캔
	20	2975 <--스캔
	20	3000 <--스캔
	20	3000 <--스캔
	30	950 <--스캔
	30	1250 <--스캔 찾음
	30	1250 <--스캔 찾음
	30	1500 <--skip
	30	1600 <--skip
	30	2850 <--skip

문제 23. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
drop index emp_deptno_sal;
```

```
create index emp_job_sal
on emp(job, sal);
```

```
select ename, job, sal, deptno
from emp
where sal = 3000;
```

```
=====
ENAME                JOB                SAL        DEPTNO
-----
FORD                  ANALYST          3000         20
SCOTT                 ANALYST          3000         20
=====
```

```
-----
| Id | Operation                | Name |
-----
|  0 | SELECT STATEMENT         |      |
|*  1 |  TABLE ACCESS FULL      | EMP  |
-----
```

```
=====
0  db block gets
4  consistent gets
0  physical reads
=====
```

튜닝후 :

```
select /*+ index_ss(emp emp_job_sal) */ ename, job, sal, deptno
from emp
where sal = 3000;
```

```
=====
ENAME                JOB                SAL        DEPTNO
-----
FORD                  ANALYST          3000         20
SCOTT                 ANALYST          3000         20
=====
```

```
-----
| Id | Operation                | Name |
-----
|  0 | SELECT STATEMENT         |      |
|  1 |  TABLE ACCESS BY INDEX ROWID | EMP  |
|*  2 |    INDEX SKIP SCAN        | EMP_JOB_SAL |
-----
```

```
=====
0  db block gets
4  consistent gets
0  physical reads
=====
```

문제 24. demobld를 돌리고 이름에 index를 거시오
그리고 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
create index emp_ename
on emp(ename);
```

```
select ename, sal, job, deptno
from emp
where ename like '%EN%'
or ename like '%IN%';
```

ENAME	SAL	JOB	DEPTNO
KING	5000	PRESIDENT	10
MARTIN	1250	SALESMAN	30
ALLEN	1600	SALESMAN	30

Id	Operation	Name
0	SELECT STATEMENT	
* 1	TABLE ACCESS FULL	EMP

```
=====
0 db block gets
4 consistent gets
0 physical reads
=====
```

튜닝후 :

- 이름에 EN 또는 IN을 포함하고 있는 사원들의 데이터의 rowid를 emp_ename 에서 빠르게 스캔해서 가져오는 쿼리를 만든다.
- 위의 쿼리를 in line view로 만들고 rowid를 연결고리로 해서 emp와 조인한다.

```
select e.ename, e.sal, e.job, e.deptno
from emp e, ( select /*+ index_ffs(emp emp_ename) */ rowid
from emp
where ename like '%EN%'
or ename like '%IN%') r
where e.rowid = r.rowid;
```

ENAME	SAL	JOB	DEPTNO
KING	5000	PRESIDENT	10
MARTIN	1250	SALESMAN	30
ALLEN	1600	SALESMAN	30

Id	Operation	Name
0	SELECT STATEMENT	
* 1	TABLE ACCESS FULL	EMP

```
=====
0 db block gets
3 consistent gets
0 physical reads
=====
```

```

@demobld
create index emp_ename
on emp(ename);

alter table emp
modify ename not null;

select /*+ rowid(e) use_nl(v e) */ e.ename, e.sal, e.job, e.deptno
from emp e, ( select /*+ no_merge index_ffs(emp emp_ename) */ rowid
              from emp
              where ename like '%EN%'
              or ename like '%|N%') r
where e.rowid = r.rowid;

```

그래서

※ 데이터를 분석해놓으면 분석할수록 14건밖에 안되는 데이터를 인덱스를 쓰는게 더 아깝다

no_merge 가 없으면 inline view를 해체한다 그래서 no merge를 써서 해체하지 못하게 한다

※ no_merge 힌트 ? in line view를 해체하지 말아라 !

	Id	Operation	Name
	0	SELECT STATEMENT	
	1	NESTED LOOPS	
	2	VIEW	
*	3	INDEX FAST FULL SCAN	EMP_ENAME
	4	TABLE ACCESS BY USER ROWID	EMP

문제 25. demobld 스크립트를 다시 돌리고 사원 테이블의 empno에 primary key제약을 걸고 unique 인덱스가 자동으로 생성되었는지 확인하시오 !

```

@demobld

alter table emp
add constraint emp_empno_pk primary key(empno);

select index_name, uniqueness
from user_indexes
where table_name = 'EMP';

```

INDEX_NAME	UNIQUENESS
EMP_EMPNO_PK	UNIQUE

문제 26. 위의 상황에서 이름에 non unique index를 걸고 아래의 SQL을 수행하면 어느 컬럼의 인덱스를 사용할 것인가 ?

```
create index emp_ename
on emp(ename);
```

```
select empno, ename, sal, job
from emp
where empno = 7788 and ename = 'SCOTT';
```

↑ ↑
인덱스 인덱스

```
=====
EMPNO ENAME                      SAL JOB
-----
7788 SCOTT                      3000 ANALYST
=====
```

```
-----
| Id | Operation                      | Name          |
-----
| 0  | SELECT STATEMENT                |               |
|* 1  | TABLE ACCESS BY INDEX ROWID    | EMP           |
|* 2  | INDEX UNIQUE SCAN                | EMP_EMPNO_PK |
-----
```

```
=====
0  db block gets
2  consistent gets
0  physical reads
=====
```

* 오라클이 우선순위를 non unique 인덱스보다 unique인덱스를 훨씬 높게 준다.

* 오라클 우선순위 표

1. rowid에 의한 데이터 액세스

:

10. full table scan

문제 27. 우리반 테이블의 이름에 unique 제약을 걸고 unique 인덱스가 생성되었는지 확인하시오 !
@demobld

```
alter table emp
add constraint emp_ename_un unique(ename);
```

```
select index_name, uniqueness
from user_indexes
where table_name = 'EMP';
```

```
=====
INDEX_NAME                      UNIQUENESS
-----
EMP_ENAME_UN                     UNIQUE
=====
```

문제 28. 사원번호가 7788번인 사원의 사원번호와 이름과 월급을 출력하는 SQL의 실행계획을 보고 index unique scan 했는지 확인하시오 !

```
select empno, ename, sal
from emp
where empno = 7788;
```

Id	Operation	Name
0	SELECT STATEMENT	
1	TABLE ACCESS BY INDEX ROWID	EMP
* 2	INDEX UNIQUE SCAN	EMP_EMPNO_PK

인덱스

```

7369 AAFAADAABAAALCxAAK
7499 AAFAADAABAAALCxAAF
7521 AAFAADAABAAALCxAAI
7566 AAFAADAABAAALCxAAD
7654 AAFAADAABAAALCxAAE
7698 AAFAADAABAAALCxAAB
7782 AAFAADAABAAALCxAAC
[ 7788 AAFAADAABAAALCxAAL ] <----- INDEX UNIQUE SCAN 은 딱 한건만 읽는다!
7839 AAFAADAABAAALCxAAA
7844 AAFAADAABAAALCxAAG
7876 AAFAADAABAAALCxAAM
7900 AAFAADAABAAALCxAAH
7902 AAFAADAABAAALCxAAJ
7934 AAFAADAABAAALCxAAN
```

문제 29. 아래의 SQL을 튜닝하시오 !

```
create index emp2_ename on emp2(ename);
```

튜닝전 :

```
select ename, age, major
from emp2
where substr(ename, 1, 1) = '김';
```

ENAME	AGE	MAJOR
김준구	27	보건행정학과
김준하	27	정보통계보험수리학과
김건휘	26	통계학과
김용원	31	컴퓨터과학과
김혜진	23	마케팅학과
김진철	33	물리학과
김용식	32	분자생물학

Id	Operation	Name
0	SELECT STATEMENT	

```
|* 1 | TABLE ACCESS FULL| EMP2 |
```

```
=====
0 db block gets
3 consistent gets
0 physical reads
=====
```

튜닝후 :

```
select /*+ index(emp2 emp2_ename) */ ename, age, major
  from emp2
 where ename like '김%';
```

```
=====
ENAME          AGE MAJOR
-----
김준구          27 보건행정학과
김준하          27 정보통계보험수리학과
김건희          26 통계학과
김용원          31 컴퓨터과학과
김혜진          23 마케팅학과
김진철          33 물리학과
김용식          32 분자생물학
=====
```

```
-----
| Id | Operation                                | Name |
-----
|  0 | SELECT STATEMENT                        |      |
|  1 | TABLE ACCESS BY INDEX ROWID| EMP2  |
|*  2 | INDEX RANGE SCAN                    | EMP2_ENAME |
-----
```

```
=====
0 db block gets
4 consistent gets
0 physical reads
=====
```

문제 30. 아래의 SQL을 튜닝하시오 !

```
create index emp2_age on emp2(age);
```

튜닝전 :

```
select ename, age, major
  from emp2
 where age like '4%';
      숫자 > 문자      문자 --> 숫자 해야하는데 % 때문에 못함
```

```
=====
ENAME          AGE MAJOR
-----
허석우          40 정보통신공학과
=====
```

```
-----
| Id | Operation                                | Name |
-----
|  0 | SELECT STATEMENT                        |      |
|*  1 | TABLE ACCESS FULL| EMP2  |
-----
```

```
=====
0 db block gets
```

```

3 consistent gets
0 physical reads
=====

```

튜닝후 :

```
create index emp2_age_func on emp2(to_char(age));
```

```
select ename, age, major
  from emp2
 where age like '4%';
```

```
=====
ENAME          AGE MAJOR
-----
허석우          40 정보통신공학과
=====

```

```

| Id | Operation                      | Name |
|----|-----|-----|
|  0 | SELECT STATEMENT                |      |
|  1 | TABLE ACCESS BY INDEX ROWID    | EMP2 |
|*  2 | INDEX RANGE SCAN                 | EMP2_AGE_FUNC |
=====

```

```

=====
0 db block gets
2 consistent gets
0 physical reads
=====

```

문제 31. (점심시간 문제) 아래의 sql을 튜닝하시오 !

전공에 통계가 포함되어져 있는 학생들의 이름과 전공을 출력하시오 !

```
create index emp2_major on emp2(major);
```

튜닝전 :

```
select ename, major
  from emp2
 where major like '%통계%';
```

```
=====
ENAME          MAJOR
-----
김준하          정보통계보험수리학과
김건희          통계학과
주소현          통계학과
이소진          통계학과
안혜진          통계학과
=====

```

```

| Id | Operation                      | Name |
|----|-----|-----|
|  0 | SELECT STATEMENT                |      |
|*  1 | TABLE ACCESS FULL              | EMP2 |
=====

```

```

=====
0 db block gets
3 consistent gets
0 physical reads
=====

```

튜닝후 :

```

select e.ename, e.major
  from emp2 e, (select /*+ no_merge index_ffs(emp2 emp2_major) */ rowid rn
                from emp2
                where major like '%통계%') r
 where e.rowid = r.rn;

```

ENAME	MAJOR
김준하	정보통계보험수리학과
김건희	통계학과
주소현	통계학과
이소진	통계학과
안혜진	통계학과

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	VIEW	
* 3	INDEX FAST FULL SCAN	EMP2_MAJOR
4	TABLE ACCESS BY USER ROWID	EMP2

```

=====
0  db block gets
6  consistent gets
0  physical reads
=====

```

문제 32. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```

select ename, sal
  from emp
 where sal = ( select max(sal) from emp);

```

ENAME	SAL
KING	5000

Id	Operation	Name
0	SELECT STATEMENT	
* 1	TABLE ACCESS FULL	EMP
2	SORT AGGREGATE	
3	TABLE ACCESS FULL	EMP

```

=====
0  db block gets
7  consistent gets
0  physical reads
=====

```

튜닝후 :

```

create index emp_sal
  on emp( sal );

```

```

select /*+ index_desc(emp emp_sal) */ ename, sal

```



```

from emp
where sal >= 0 and rownum = 1;
=====
ENAME                      SAL
-----
KING                      5000
=====

| Id | Operation
-----
| 0 | SELECT STATEMENT
|* 1 | COUNT STOPKEY
| 2 | TABLE ACCESS BY INDEX ROWID
|* 3 | INDEX RANGE SCAN DESCENDING
-----

=====
0 db block gets
2 consistent gets
0 physical reads
=====

```

문제 33. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```

select ename, hiredate
from emp
where hiredate = (select min(hiredate) from emp);
=====
ENAME                      HIREDATE
-----
SMITH                      80/12/09
=====

| Id | Operation          | Name |
-----
| 0 | SELECT STATEMENT   |      |
|* 1 | TABLE ACCESS FULL | EMP  |
| 2 | SORT AGGREGATE     |      |
| 3 | TABLE ACCESS FULL | EMP  |
-----

=====
0 db block gets
7 consistent gets
0 physical reads
=====

```

튜닝후 :

```

create index emp_hiredate
on emp(hiredate);

select /*+ index_asc(emp emp_hiredate) */ ename, hiredate
from emp
where hiredate < to_date ( '9999/12/31', 'rrrr/mm/dd' )
and rownum = 1;
=====
ENAME                      HIREDATE
-----
SMITH                      80/12/09
=====

```

Id	Operation
0	SELECT STATEMENT
* 1	COUNT STOPKEY
2	TABLE ACCESS BY INDEX ROWID
* 3	INDEX RANGE SCAN DESCENDING

```

=====
0  db block gets
2  consistent gets
0  physical reads
=====

```

문제 34. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```

select ename, sal
  from emp e
 where sal > (select avg(sal)
               from emp s
              where s.deptno = e.deptno);

```

ENAME	SAL
KING	5000
BLAKE	2850
JONES	2975
ALLEN	1600
FORD	3000
SCOTT	3000

Id	Operation	Name
0	SELECT STATEMENT	
* 1	FILTER	
2	TABLE ACCESS FULL	EMP
3	SORT AGGREGATE	
* 4	TABLE ACCESS FULL	EMP

```

=====
0  db block gets
13 consistent gets
0  physical reads
=====

```

튜닝전 :

```

select ename, sal
  from (select ename, sal, avg(sal) over ( partition by deptno) avg
        from emp)
 where sal > avg;

```

ENAME	SAL
KING	5000
SCOTT	3000
FORD	3000
JONES	2975
ALLEN	1600

BLAKE 2850

=====

	Id	Operation	Name
	0	SELECT STATEMENT	
*	1	VIEW	
	2	WINDOW SORT	
	3	TABLE ACCESS FULL	EMP

=====

0 db block gets
3 consistent gets
0 physical reads

=====

※ inline view는 TABLE ACCESS 를 1번 한다.

문제 35. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
select ename, sal, job
  from emp e
 where 4 <= (select count(*)
             from emp s
             where s.job = e.job);
```

=====

ENAME	SAL	JOB
MARTIN	1250	SALESMAN
ALLEN	1600	SALESMAN
TURNER	1500	SALESMAN
JAMES	950	CLERK
WARD	1250	SALESMAN
SMITH	800	CLERK
ADAMS	1100	CLERK
MILLER	1300	CLERK

=====

	Id	Operation	Name
	0	SELECT STATEMENT	
*	1	FILTER	
	2	TABLE ACCESS FULL	EMP
	3	SORT AGGREGATE	
*	4	TABLE ACCESS FULL	EMP

=====

0 db block gets
19 consistent gets
0 physical reads

=====

튜닝후 :

```
select ename, sal, job
  from (select ename, sal, job, count(*) over (partition by job) cnt
        from emp)
 where cnt >= 4;
```

=====

ENAME	SAL	JOB
SMITH	800	CLERK
MILLER	1300	CLERK
JAMES	950	CLERK
ADAMS	1100	CLERK
WARD	1250	SALESMAN
ALLEN	1600	SALESMAN
MARTIN	1250	SALESMAN
TURNER	1500	SALESMAN

Id	Operation	Name
0	SELECT STATEMENT	
* 1	VIEW	
2	WINDOW SORT	
3	TABLE ACCESS FULL	EMP


```

=====
0  db block gets
3  consistent gets
0  physical reads
=====

```

문제 36. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```

select job, sum(sal)
  from emp
 group by job
 union
select null as job, sum(sal)
  from emp;

```

JOB	SUM(SAL)
ANALYST	6000
CLERK	4150
MANAGER	8275
PRESIDENT	5000
SALESMAN	5600
	29025

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT UNIQUE	
2	UNION-ALL	
3	HASH GROUP BY	
4	TABLE ACCESS FULL	EMP
5	SORT AGGREGATE	
6	TABLE ACCESS FULL	EMP


```

=====
0  db block gets
6  consistent gets
0  physical reads

```

튜닝후 :

```
select job, sum(sal)
  from emp
 group by rollup(job);
```

```
=====
JOB                                SUM(SAL)
-----
ANALYST                            6000
CLERK                              4150
MANAGER                            8275
PRESIDENT                          5000
SALESMAN                           5600
                                     29025
=====
```

```
-----
| Id | Operation                      | Name |
-----
|  0 | SELECT STATEMENT                |      |
|  1 |  SORT GROUP BY ROLLUP           |      |
|  2 |    TABLE ACCESS FULL          | EMP  |
-----
```

```
=====
0  db block gets
3  consistent gets
0  physical reads
=====
```

문제 37. 아래의 SQL의 결과를 union all로 변경하시오 !

튜닝전 :

```
select deptno, job, avg(sal)
  from emp
 group by grouping sets( (deptno), (job) );
```

```
=====
DEPTNO JOB                                AVG(SAL)
-----
      30                                1566.66667
      20                                 2175
      10                                2916.66667
      SALESMAN                          1400
      CLERK                             1037.5
      PRESIDENT                          5000
      MANAGER                           2758.33333
      ANALYST                            3000
=====
```

```
-----
| Id | Operation                      | Name |
-----
|  0 | SELECT STATEMENT                |      |
|  1 |  TEMP TABLE TRANSFORMATION     |      |
|  2 |    LOAD AS SELECT               | SYS_TEMP_OFD9D6602_2CAC9C |
|  3 |      TABLE ACCESS FULL         | EMP  |
|  4 |    LOAD AS SELECT               | SYS_TEMP_OFD9D6603_2CAC9C |
|  5 |      HASH GROUP BY              |      |
|  6 |        TABLE ACCESS FULL       | SYS_TEMP_OFD9D6602_2CAC9C |
|  7 |          LOAD AS SELECT          | SYS_TEMP_OFD9D6603_2CAC9C |
-----
```

8	HASH GROUP BY	
9	TABLE ACCESS FULL	SYS_TEMP_0FD9D6602_2CAC9C
10	VIEW	
11	TABLE ACCESS FULL	SYS_TEMP_0FD9D6603_2CAC9C

```
=====
28  db block gets
105 consistent gets
3   physical reads
=====
```

```
select deptno, null as job, avg(sal)
  from emp
  group by deptno
union all
select null as deptno, job, avg(sal)
  from emp
  group by job;
```

DEPTNO	JOB	AVG(SAL)
30		1566.66667
20		2175
10		2916.66667
	SALESMAN	1400
	CLERK	1037.5
	PRESIDENT	5000
	MANAGER	2758.33333
	ANALYST	3000

Id	Operation	Name
0	SELECT STATEMENT	
1	UNION-ALL	
2	HASH GROUP BY	
3	TABLE ACCESS FULL	EMP
4	HASH GROUP BY	
5	TABLE ACCESS FULL	EMP

```
=====
0  db block gets
18 consistent gets
0  physical reads
=====
```

튜닝후 :

```
select /*+ expand_gset_to_union */ deptno, job, avg(sal)
  from emp
  group by grouping sets( (deptno), (job) );
```

※ 설명 : expand_gset_to_union 힌트는?

오라클 옵티마이저에게 SQL을 네가 union all로 작성해라라고 명령을 내리는 힌트

DEPTNO	JOB	AVG(SAL)
30		1566.66667
20		2175

```

10          2916.66667
SALESMAN    1400
CLERK       1037.5
PRESIDENT   5000
MANAGER     2758.33333
ANALYST     3000

```

=====				
Id	Operation	Name		

0	SELECT STATEMENT			
1	VIEW			
2	UNION-ALL			
3	HASH GROUP BY			
4	TABLE ACCESS FULL	EMP		
5	HASH GROUP BY			
6	TABLE ACCESS FULL	EMP		

```

=====
0  db block gets
6  consistent gets
0  physical reads
=====

```

문제 38. 이름과 부서위치를 출력하는 조인문장의 실행계획을 보고 emp테이블을 먼저 읽고 dept랑 조인했는지 dept테이블을 먼저 읽고 emp랑 조인했는지 알아내시오 !

@demobld

```

select e.ename, d.deptno
  from emp e, dept d
 where e.deptno = d.deptno;

```

Id	Operation	Name		

0	SELECT STATEMENT			4
* 1	HASH JOIN			3
2	TABLE ACCESS FULL	DEPT		1
3	TABLE ACCESS FULL	EMP		2

```

=====
0  db block gets
21 consistent gets
0  physical reads
=====

```

※ 읽는 순서가 안쪽에서부터 읽는다,
 ※ 성능을 따져보면 dept를 먼저 읽는게 빠를까? emp를 먼저 읽는게 빠를까?
 테이블이 작은걸 먼저 읽는게 빠르다
 그래서 옵티마이저가 똑똑하게 dept를 먼저 읽은 것이다.

```

dept -----> emp 0
emp -----> dept X

```

* nested loop join을 사용하려면 ?

```

select /*+ use_nl(d e) */ e.ename, d.deptno
  from emp e, dept d
 where e.deptno = d.deptno;

```

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	TABLE ACCESS FULL	DEPT
* 3	TABLE ACCESS FULL	EMP

```

=====
0  db block gets
17  consistent gets
0  physical reads
=====

```

* 조인 하는 순서를 변경하는 힌트

1. ordered : from 절에서 기술한 순서대로 조인하겠다.

```

select /*+ ordered use_nl(d e) */ e.ename, d.deptno
  from emp e, dept d
 where e.deptno = d.deptno;

```

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	TABLE ACCESS FULL	EMP
* 3	TABLE ACCESS FULL	DEPT

```

=====
0  db block gets
47  consistent gets
0  physical reads
=====

```

```

select /*+ ordered use_nl(d e) */ e.ename, d.deptno
  from dept d, emp e
 where e.deptno = d.deptno;

```

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	TABLE ACCESS FULL	DEPT
* 3	TABLE ACCESS FULL	EMP

```

=====
0  db block gets
17  consistent gets
0  physical reads
=====

```

2. leading : leading 힌트 안에 쓴 테이블 순서대로 조인하겠다.

```

select /*+ leading(e d) use_nl(d e) */ e.ename, d.deptno
  from emp e, dept d
 where e.deptno = d.deptno;

```


문제 39. emp와 salgrade와 dept를 조인해서 이름과 월급과 부서위치, 급여등급을 출력하시오 !

```
select e.ename, e.sal, d.loc, s.grade
from emp e, dept d, salgrade s
where e.deptno = d.deptno
and e.sal between s.losal and s.hisal;
```

	Id	Operation	Name	
	0	SELECT STATEMENT		
*	1	HASH JOIN		
	2	MERGE JOIN CARTESIAN		
	3	TABLE ACCESS FULL	DEPT	
	4	BUFFER SORT		
	5	TABLE ACCESS FULL	SALGRADE	
	6	TABLE ACCESS FULL	EMP	

```
=====
0  db block gets
32  consistent gets
0  physical reads
=====
```

문제 40. 위의 조인문장의 조인순서와 조인 방법을 아래의 방법으로 수행하시오 !

조인 순서 : salgrade ---> emp ---> dept

조인 방법 : nested loop join nested loop join

```
select /*+ leading(s e d) use_nl(e) use_nl(d) */ e.ename, e.sal, d.loc, s.grade
from emp e, dept d, salgrade s
where e.deptno = d.deptno
and e.sal between s.losal and s.hisal;
```

	Id	Operation	Name	
	0	SELECT STATEMENT		5
	1	NESTED LOOPS		4
	2	NESTED LOOPS		3
	3	TABLE ACCESS FULL	SALGRADE	1
*	4	TABLE ACCESS FULL	EMP	2
*	5	TABLE ACCESS FULL	DEPT	4

```
=====
0  db block gets
75  consistent gets
0  physical reads
=====
```

※ 힌트 해석 : salgrade, emp 그리고 dept 순서로 읽는데[leading(s e d)] 첫번째로 읽은것과 emp테이블 을 nested loop join[use_nl(e)] 하고
조인한 테이블과 dept테이블을 nested loop join[use_nl(d)] 한다.

문제 41. 위의 SQL이 아래와 같은 조인순서로 실행되게 하시오 !

조인 순서 : DEPT ---> EMP ---> SALGRADE

↑ ↑

조인 방법 : nested loop join nested loop join

```
select /*+ leading(d e s) use_nl(e) use_nl(s) */ e.ename, e.sal, d.loc, s.grade
  from emp e, dept d, salgrade s
 where e.deptno = d.deptno
       and e.sal between s.losal and s.hisal;
```

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	NESTED LOOPS	
3	TABLE ACCESS FULL	DEPT
* 4	TABLE ACCESS FULL	EMP
* 5	TABLE ACCESS FULL	SALGRADE

```
=====
0  db block gets
59 consistent gets
0  physical reads
=====
```

★ 현업버전

```
=====
create table sales600
as
select *
  from sales;
```

```
create table customers600
as
select *
  from customers;
```

```
=====
SELECT /*+ leading(s c) use_nl(c) */ COUNT(*)
  FROM sales600 s, customers600 c
 WHERE s.cust_id = c.cust_id
    AND c.country_id = 52790
    AND s.time_id BETWEEN TO_DATE('1999/01/01','YYYY/MM/DD')
                      AND TO_DATE('1999/12/31','YYYY/MM/DD');
  ※ 너무 오래걸려서 스톱함 (ctrl + c)
```

```
SELECT /*+ leading(c s) use_nl(s) */ COUNT(*)
  FROM sales600 s, customers600 c
 WHERE s.cust_id = c.cust_id
    AND c.country_id = 52790
    AND s.time_id BETWEEN TO_DATE('1999/01/01','YYYY/MM/DD')
                      AND TO_DATE('1999/12/31','YYYY/MM/DD');
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	
0	SELECT STATEMENT		1	48	19M (1)	63:31:43	
1	SORT AGGREGATE		1	48			
2	NESTED LOOPS		233K	10M	19M (1)	63:31:43	
* 3	TABLE ACCESS FULL	CUSTOMERS600	15810	401K	417 (1)	00:00:06	
* 4	TABLE ACCESS FULL	SALES600	15	330	1205 (1)	00:00:15	

```
=====
0 db block gets
82137735 consistent gets
0 physical reads
=====
```

문제 42. (오늘의 마지막 문제) 아래의 SQL을 튜닝하시오 !
(무조건 nested loop 조인으로 수행하되 가장 좋은 조인순서를 결정하시오 !)

```
create table sales100 as select * from sales;
create table times100 as select * from times;
create table products100 as select * from products;
```

```
튜닝 전 :
select /*+ leading(s t p) use_nl(t) use_nl(p) */
p.prod_name, t.CALENDAR_YEAR, sum(s.amount_sold)
from sales100 s, times100 t, products100 p
where s.time_id = t.time_id
and s.prod_id = p.prod_id
and t.CALENDAR_YEAR in (2000,2001)
and p.prod_name like 'Deluxe%'
group by p.prod_name, t.calendar_year;
```

```
답1 :
select /*+ leading(p s t) use_nl(s) use_nl(t) */
p.prod_name,
t.CALENDAR_YEAR,
sum(s.amount_sold)
from sales100 s, /* 데이터 918,843건 */
times100 t, /* 데이터 1,826건 */
products100 p /* 데이터 72건 */
where s.time_id = t.time_id /* s,t 연결고리 */
and s.prod_id = p.prod_id /* s,p 연결고리 */
and t.CALENDAR_YEAR in (2000,2001) /* times100 731건 */
and p.prod_name like 'Deluxe%' /* products100 1건 */
group by p.prod_name, t.calendar_year;
```

/* 테이블 데이터 건수로만 따지면 p-t-s가 가장 빠를것 같지만 조인연결고리가 훼손된다.
테이블의 연결고리가 t-s-p이기 때문에 연결고리를 훼손하지 않는 선에서
p-s-t(72건-918,843건-1,826건)순서로 액세스하는 것이 가장 빠를 것으로 예상된다. */;

```
=====
Deluxe Mouse 2001 142334.42
Deluxe Mouse 2000 53224.73
=====
경 과: 00:00:02.96
=====
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		19750	2835K	323K (1)	01:04:41
1	HASH GROUP BY		19750	2835K	323K (1)	01:04:41
2	NESTED LOOPS		19750	2835K	323K (1)	01:04:41
3	NESTED LOOPS		19750	2410K	1383 (1)	00:00:17
* 4	TABLE ACCESS FULL	PRODUCTS100	1	90	2 (0)	00:00:01
* 5	TABLE ACCESS FULL	SALES100	19750	675K	1381 (1)	00:00:17
* 6	TABLE ACCESS FULL	TIMES100	1	22	16 (0)	00:00:01


```

=====
      0  db block gets
813943  consistent gets
   5060  physical reads
=====

```

선생님 설명

1. products 테이블에서 p.prod_name like 'Deruxe%' 조건의 데이터 1건을 찾아낸다.
2. 1건의 prod_id 47번을 sales100테이블에 조인시도를 한다(1건밖에 없으므로 1번만 조인시도 한다.)
3. prod_id 47번을 sales100테이블에서 12837건을 찾아낸다.
4. prod_id 12837건을 times100 테이블로 조인시도를 한다 (조인 시도가 12837 번)
5. times100 테이블로 조인 시도한 12837건중에 CALENDAR_YEAR in (2000,2001)조건에 만족한 것만 결과로 출력된다.

중첩 루프 조인 -----> 이중 루프문 프로그램

```

loop .....
  loop.....
end loop.....
loop.....

```

문제 43. sales100 테이블의 prod_id에 인덱스를 걸면 더 속도가 빨라지는지 확인하시오 !

```

create index sales100_prod_id
on sales100(prod_id);

```

```

create index times100_time_id
on times100(time_id);

```

```

select /*+ leading(p s t) use_nl(s) use_nl(t) */
      p.prod_name,
      t.CALENDAR_YEAR,
      sum(s.amount_sold)
from    sales100      s,
        times100      t,
        products100   p
where   s.time_id = t.time_id
and     s.prod_id = p.prod_id
and     t.CALENDAR_YEAR in (2000,2001)

```

```

and p.prod_name like 'Deluxe%'
group by p.prod_name, t.calendar_year;

```

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
2	NESTED LOOPS	
3	NESTED LOOPS	
4	NESTED LOOPS	
* 5	TABLE ACCESS FULL	PRODUCTS100
6	TABLE ACCESS BY INDEX ROWID	SALES100
* 7	INDEX RANGE SCAN	SALES100_PROD_ID
* 8	INDEX RANGE SCAN	TIMES100_TIME_ID
* 9	TABLE ACCESS BY INDEX ROWID	TIMES100

경 과: 00:00:00.06

```

create index sales100_time_id
on sales100(time_id);

```

```

create index products100_prod_id
on products100(prod_id);

```

문제 44. 아래의 sql을 튜닝하시오 !

(조인 방법은 무조건 nested loop join으로 하고 조인순서는 알아서 결정하고, 인덱스도 알아서 생성하시오 !)

튜닝전 :

```

SELECT /*+ leading(c s) use_nl(s) */ COUNT(*)
FROM sales600 s, customers600 c
WHERE s.cust_id = c.cust_id
AND c.country_id = 52790
AND s.time_id BETWEEN TO_DATE('1999/01/01','YYYY/MM/DD')
AND TO_DATE('1999/12/31','YYYY/MM/DD');

```

튜닝후 :

```

create index sales600_cust_id
on sales600(cust_id);

```

```

create index customers600_cust_id
on customers600(cust_id);

```

```

SELECT /*+ leading(c s) use_nl(s) */ COUNT(*)
FROM sales600 s, customers600 c
WHERE s.cust_id = c.cust_id
AND c.country_id = 52790
AND s.time_id BETWEEN TO_DATE('1999/01/01','YYYY/MM/DD')
AND TO_DATE('1999/12/31','YYYY/MM/DD');

```

```

=====
COUNT(*)
=====
141806

```

```
=====
경  과: 00:00:01.09
=====
```

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT AGGREGATE	
2	NESTED LOOPS	
3	NESTED LOOPS	
* 4	TABLE ACCESS FULL	CUSTOMERS600
* 5	INDEX RANGE SCAN	SALES600_CUST_ID
* 6	TABLE ACCESS BY INDEX ROWID	SALES600

```
=====
      0  db block gets
455661  consistent gets
      1736  physical reads
=====
```

문제 45. 아래의 SQL의 조인방법은 무조건 nested loop join으로 하되 조인 순서를 결정하고 인덱스도 알아서 생성하시오 !

튜닝전 :

```
select e.ename, e.sal, d.loc, e.deptno
  from emp e, dept d                /* emp 14건 dept 4건*/
 where e.deptno = d.deptno
    and e.job = 'SALESMAN'          /* 4건 */
    and d.loc = 'CHICAGO';          /* 1건 */
```

ENAME	SAL	LOC	DEPTNO
MARTIN	1250	CHICAGO	30
ALLEN	1600	CHICAGO	30
TURNER	1500	CHICAGO	30
WARD	1250	CHICAGO	30

```
=====
경  과: 00:00:00.03
=====
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	60	4 (0)	00:00:01
1	NESTED LOOPS		1	60	4 (0)	00:00:01
* 2	TABLE ACCESS FULL	DEPT	1	21	2 (0)	00:00:01
* 3	TABLE ACCESS FULL	EMP	1	39	2 (0)	00:00:01

```
=====
      0  db block gets
      16  consistent gets
      0  physical reads
=====
```

튜닝후 :

```
create index dept_deptno
  on dept(deptno);
create index emp_deptno
  on emp(deptno);
```

```

select /*+ leading(d e) use_nl(e) */
       e.ename, e.sal, d.loc, e.deptno
  from emp e, dept d
 where e.deptno = d.deptno
       and e.job = 'SALESMAN'
       and d.loc = 'CHICAGO';

```

```

=====
ENAME                SAL LOC                DEPTNO
-----
MARTIN                1250 CHICAGO                30
ALLEN                  1600 CHICAGO                30
TURNER                 1500 CHICAGO                30
WARD                   1250 CHICAGO                30
=====

```

경 과: 00:00:00.03

```

-----
| Id | Operation                                | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----
|  0 | SELECT STATEMENT                        |               |      1 |    60 |      3 (0)| 00:00:01 |
|  1 |   NESTED LOOPS                          |               |      |      |           |          |
|  2 |    NESTED LOOPS                          |               |      1 |    60 |      3 (0)| 00:00:01 |
|*  3 |     TABLE ACCESS FULL                   | DEPT          |      1 |    21 |      2 (0)| 00:00:01 |
|*  4 |      INDEX RANGE SCAN                     | EMP_DEPTNO    |      5 |      |      0 (0)| 00:00:01 |
|*  5 |       TABLE ACCESS BY INDEX ROWID      | EMP           |      1 |    39 |      1 (0)| 00:00:01 |
-----

```

```

=====
0  db block gets
18 consistent gets
0  physical reads
=====

```

문제 46. 아래의 SQL을 조금 더 튜닝하시오 !

튜닝전 :

```

SELECT /*+ leading(c s) use_nl(s) */ COUNT(*)
  FROM sales600 s, customers600 c
 WHERE s.cust_id = c.cust_id
       AND c.country_id = 52790
       AND s.time_id BETWEEN TO_DATE('1999/01/01','YYYY/MM/DD')
                          AND TO_DATE('1999/12/31','YYYY/MM/DD');

```

COUNT(*)

141806

경 과: 00:00:00.43

```

-----
| Id | Operation                                | Name          |
-----
|  0 | SELECT STATEMENT                        |               |
|  1 |   SORT AGGREGATE                        |               |
|  2 |    NESTED LOOPS                          |               |
|  3 |     NESTED LOOPS                          |               |
|*  4 |      TABLE ACCESS FULL                   | CUSTOMERS600  |
|*  5 |       INDEX RANGE SCAN                     | SALES600_CUST_ID |
|*  6 |        TABLE ACCESS BY INDEX ROWID      | SALES600       |
-----

```

```
0 db block gets
456976 consistent gets
0 physical reads
```

튜닝후 :

```
create index customers600_country_id
on customers600(country_id);
```

```
create index sales600_time_id
on sales600(time_id);
```

```
SELECT /*+ leading(c s) use_nl(s) */ COUNT(*)
FROM sales600 s, customers600 c
WHERE s.cust_id = c.cust_id
AND c.country_id = 52790
AND s.time_id BETWEEN TO_DATE('1999/01/01','YYYY/MM/DD')
AND TO_DATE('1999/12/31','YYYY/MM/DD');
```

COUNT(*)

141806

경 과: 00:00:00.48

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT AGGREGATE	
2	NESTED LOOPS	
3	NESTED LOOPS	
4	TABLE ACCESS BY INDEX ROWID	CUSTOMERS600
* 5	INDEX RANGE SCAN	CUSTOMERS600_COUNTRY_ID
* 6	INDEX RANGE SCAN	SALES600_CUST_ID
* 7	TABLE ACCESS BY INDEX ROWID	SALES600

```
0 db block gets
456970 consistent gets
42 physical reads
```

문제 47. 위의 SQL을 hash join으로 변경하시오 !

답1

```
SELECT /*+ leading(c s) use_hash(s) */ COUNT(*)
FROM sales600 s, customers600 c
WHERE s.cust_id = c.cust_id
AND c.country_id = 52790
AND s.time_id BETWEEN TO_DATE('1999/01/01','YYYY/MM/DD')
AND TO_DATE('1999/12/31','YYYY/MM/DD');
```

COUNT(*)

141806

경 과: 00:00:00.21

Id	Operation	Name	실행순서
0	SELECT STATEMENT		7
1	SORT AGGREGATE		6
* 2	HASH JOIN		5
3	TABLE ACCESS BY INDEX ROWID	CUSTOMERS600	2
* 4	INDEX RANGE SCAN	CUSTOMERS600_COUNTRY_ID	1
5	TABLE ACCESS BY INDEX ROWID	SALES600	4
* 6	INDEX RANGE SCAN	SALES600_TIME_ID	3

0 db block gets
15242 consistent gets
658 physical reads

답2

```
SELECT /*+ leading(c s) use_hash(s) full(c) full(s) */ COUNT(*)
  FROM sales600 s, customers600 c
 WHERE s.cust_id = c.cust_id
    AND c.country_id = 52790
    AND s.time_id BETWEEN TO_DATE('1999/01/01','YYYY/MM/DD')
                        AND TO_DATE('1999/12/31','YYYY/MM/DD');
```

COUNT(*)

141806

경 과: 00:00:00.07

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT AGGREGATE	
* 2	HASH JOIN	
* 3	TABLE ACCESS FULL	CUSTOMERS600
* 4	TABLE ACCESS FULL	SALES600

0 db block gets
6096 consistent gets
0 physical reads

문제 48. 아래의 sql 을 hash조인으로 수행되게 한 후 최대한 줄인 블록의 갯수로 검사 받으세요 !

튜닝 전 :

```
select /*+ leading(p s t) use_nl(s) use_nl(t) */
      p.prod_name,
      t.CALENDAR_YEAR,
      sum(s.amount_sold)
  from   sales100    s,
         times100    t,
         products100 p
 where  s.time_id = t.time_id
    and s.prod_id = p.prod_id
    and t.CALENDAR_YEAR in (2000,2001)
    and p.prod_name like 'Deluxe%'
 group by p.prod_name, t.calendar_year;
```

/* 데이터 918,843건 */
/* 데이터 1,826건 */
/* 데이터 72건 */
/* s,t 연결고리 */
/* s,p 연결고리 */
/* times100 731건 */
/* products100 1건 */

튜닝 후

```
select /*+ leading(p s t) use_hash(t) full(t)
      index(products100 products100_prod_id) */
      p.prod_name,
```

```

        t.CALENDAR_YEAR,
        sum(s.amount_sold)
from   sales100    s,
        times100   t,
        products100 p
where  s.time_id = t.time_id
      and s.prod_id = p.prod_id
      and t.CALENDAR_YEAR in (2000,2001)
      and p.prod_name like 'Deluxe%'
group by p.prod_name, t.calendar_year;

```

문제 49. 아래와 같이 실행계획이 나오게 하시오 !

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
* 2	HASH JOIN	
* 3	TABLE ACCESS FULL	PRODUCTS100
* 4	HASH JOIN	
* 5	TABLE ACCESS FULL	TIMES100
6	TABLE ACCESS FULL	SALES100

```

select /*+ leading(t s p) use_hash(t) full(t) full(p) */
      p.prod_name,
      t.CALENDAR_YEAR,
      sum(s.amount_sold)
from   sales100    s,
        times100   t,
        products100 p
where  s.time_id = t.time_id
      and s.prod_id = p.prod_id
      and t.CALENDAR_YEAR in (2000,2001)
      and p.prod_name like 'Deluxe%'
group by p.prod_name, t.calendar_year;

```

Deluxe Mouse

142334.42

Deluxe Mouse

53224.73

경과: 00:00:00.28

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
* 2	HASH JOIN	
* 3	TABLE ACCESS FULL	PRODUCTS100
* 4	HASH JOIN	
* 5	TABLE ACCESS FULL	TIMES100
6	TABLE ACCESS FULL	SALES100

```

0      db block gets
5279   consistent gets
5060   physical reads

```

문제 50. 아래와 같이 실행계획이 나오게 하시오 !

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
* 2	HASH JOIN	
* 3	HASH JOIN	
* 4	TABLE ACCESS FULL	TIMES100
5	TABLE ACCESS FULL	SALES100
* 6	TABLE ACCESS FULL	PRODUCTS100

```

select /*+ leading(t s p) use_hash(p) full(s) full(p) full(t)
        no_swap_join_inputs(p) */
        p.prod_name,
        t.CALENDAR_YEAR,
        sum(s.amount_sold)
from   sales100   s,
        times100   t,
        products100 p
where  s.time_id = t.time_id
and    s.prod_id = p.prod_id
and    t.CALENDAR_YEAR in (2000,2001)
and    p.prod_name like 'Deluxe%'
group by p.prod_name, t.calendar_year;

```

Deluxe Mouse

142334.42

Deluxe Mouse

53224.73

경 과: 00:00:00.28

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
* 2	HASH JOIN	
* 3	HASH JOIN	
* 4	TABLE ACCESS FULL	TIMES100
5	TABLE ACCESS FULL	SALES100
* 6	TABLE ACCESS FULL	PRODUCTS100

```

0  db block gets
5279 consistent gets
5277 physical reads

```

문제 51. 아래와 같이 실행계획이 출력되게 하시오 !

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
* 2	HASH JOIN	
* 3	TABLE ACCESS FULL	PRODUCTS100
* 4	HASH JOIN	
5	TABLE ACCESS FULL	SALES100

|* 6 | TABLE ACCESS FULL| TIMES100 |

```

select /*+ leading(s t p) use_hash(t) use_hash(p) full(s) full(p) full(t)
        swap_join_inputs(p) */
        p.prod_name,
        t.CALENDAR_YEAR,
        sum(s.amount_sold)
from   sales100   s,
        times100   t,
        products100 p
where  s.time_id = t.time_id
and    s.prod_id = p.prod_id
and    t.CALENDAR_YEAR in (2000,2001)
and    p.prod_name like 'Deluxe%'
group by p.prod_name, t.calendar_year;

```

Deluxe Mouse

142334.42

Deluxe Mouse

53224.73

경 과: 00:00:00.35

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
* 2	HASH JOIN	
* 3	TABLE ACCESS FULL	PRODUCTS100
* 4	HASH JOIN	
5	TABLE ACCESS FULL	SALES100
* 6	TABLE ACCESS FULL	TIMES100

0 db block gets
 5279 consistent gets
 5060 physical reads

문제 52. emp와 salgrade 테이블을 조인해서 이름과 월급과 급여등급(grade)을 출력하시오 !

```

select e.ename, e.sal, s.grade
from emp e, salgrade s
where e.sal between s.losal and s.hisal;

```

문제 53. 위의 SQL의 실행계획을 해쉬조인으로 수행되게 하시오 !

```

select /*+ leading(s e) use_hash(e) full(s) full(e)
        swap_join_inputs(s) */
        e.ename, e.sal, s.grade
from emp e, salgrade s
where e.sal between s.losal and s.hisal;

```

※ 해쉬조인은 조인의 연결고리가 =(이퀄) 조건일 때만 가능하다.

문제 54. 아래의 SQL을 작성하는데 조인 순서와 조인 방법이 아래와 같이 되게 하시오 !

조인 순서 : dept ----> emp ----> salgrade
조인 방법 : 해쉬조인 nl조인

```
select /*+ leading(d e s) use_hash(e) use_nl(s) full(d) full(e) */
       e.ename, d.loc, e.sal, s.grade
  from emp e, dept d, salgrade s
 where e.deptno = d.deptno
       and e.sal between s.losal and s.hisal;
```

ENAME	LOC	SAL	GRADE
KING	NEW YORK	5000	5
BLAKE	CHICAGO	2850	4
CLARK	NEW YORK	2450	4
JONES	DALLAS	2975	4
MARTIN	CHICAGO	1250	2
ALLEN	CHICAGO	1600	3
TURNER	CHICAGO	1500	3
JAMES	CHICAGO	950	1
WARD	CHICAGO	1250	2
FORD	DALLAS	3000	4
SMITH	DALLAS	800	1
SCOTT	DALLAS	3000	4
ADAMS	DALLAS	1100	1
MILLER	NEW YORK	1300	2

경과: 00:00:00.06

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
0	SELECT STATEMENT		1	93	11 (10)
1	NESTED LOOPS		1	93	11 (10)
* 2	HASH JOIN		14	756	5 (20)
3	TABLE ACCESS FULL	DEPT	4	84	2 (0)
4	TABLE ACCESS FULL	EMP	14	462	2 (0)
* 5	TABLE ACCESS FULL	SALGRADE	1	39	0 (0)

63 consistent gets
0 physical reads
0 redo size

문제 55. 아래의 SQL의 조인순서와 조인방법을 아래와 같이 하시오 !

조인순서 : times ----> sales ----> products
조인방법 : 해쉬 NL

```
select /*+ leading(t s p) use_hash(s) use_nl(p) full(s) full(t)
       swap_join_inputs(t) */
       p.prod_name,
       t.CALENDAR_YEAR,
       sum(s.amount_sold)
  from   sales100 s,
         times100 t,
         products100 p
 where  s.time_id = t.time_id
       and s.prod_id = p.prod_id
```

```

and t.CALENDAR_YEAR in (2000,2001)
and p.prod_name like 'Deluxe%'
group by p.prod_name, t.calendar_year;

```

```

Deluxe Mouse
142334.42
Deluxe Mouse
53224.73

```

경 과: 00:00:04.33

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		19750	2835K	515K (1)	01:43:10
1	HASH GROUP BY		19750	2835K	515K (1)	01:43:10
2	NESTED LOOPS		19750	2835K	515K (1)	01:43:10
* 3	HASH JOIN		947K	51M	1402 (1)	00:00:17
* 4	TABLE ACCESS FULL	TIMES100	731	16082	18 (0)	00:00:01
5	TABLE ACCESS FULL	SALES100	947K	31M	1381 (1)	00:00:17
* 6	TABLE ACCESS FULL	PRODUCTS100	1	90	1 (0)	00:00:01

```

0 db block gets
1973531 consistent gets
5060 physical reads

```

문제 56. 아래와 같이 실행계획이 나오게 하시오 !

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
* 2	HASH JOIN	
* 3	TABLE ACCESS FULL	TIMES100
4	NESTED LOOPS	
5	NESTED LOOPS	
* 6	TABLE ACCESS FULL	PRODUCTS100
* 7	INDEX RANGE SCAN	SALES100_PROD_ID
8	TABLE ACCESS BY INDEX ROWID	SALES100

```

select /*+ leading(p s t) use_nl(s) use_hash(p) full(p s) full(t)
        swap_join_inputs(p)
        index(sales100 sales100_prod_id) */
        p.prod_name,
        t.CALENDAR_YEAR,
        sum(s.amount_sold)
from   sales100    s,
        times100    t,
        products100 p
where  s.time_id = t.time_id
and    s.prod_id = p.prod_id
and    t.CALENDAR_YEAR in (2000,2001)
and    p.prod_name like 'Deluxe%'
group by p.prod_name, t.calendar_year;

```

```

Deluxe Mouse
142334.42
Deluxe Mouse

```

53224.73

경 과: 00:00:00.04

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
* 2	HASH JOIN	
* 3	TABLE ACCESS FULL	TIMES100
4	NESTED LOOPS	
5	NESTED LOOPS	
* 6	TABLE ACCESS FULL	PRODUCTS100
* 7	INDEX RANGE SCAN	SALES100_PROD_ID
8	TABLE ACCESS BY INDEX ROWID	SALES100

0 db block gets
199 consistent gets
0 physical reads

```
select e.ename, d.loc  
  from emp e, dept d  
 where e.deptno(+) = d.deptno;
```

문제 57. 위의 조인문의 조인순서와 조인방법을 아래와 같이 되게 하시오 !

조인 순서 : dept ----> emp
조인 방법 : 해쉬조인

```
select /*+ leading(d e) use_hash(e) full(d) full(e) */  
      e.ename, d.loc  
  from emp e, dept d  
 where e.deptno(+) = d.deptno;
```

경 과: 00:00:00.03

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	574	5 (20)	00:00:01
* 1	HASH JOIN OUTER		14	574	5 (20)	00:00:01
2	TABLE ACCESS FULL	DEPT	4	84	2 (0)	00:00:01
3	TABLE ACCESS FULL	EMP	14	280	2 (0)	00:00:01

0 db block gets
17 consistent gets
0 physical reads

문제 58. 아래의 outer join 의 조인 순서와 조인 방법을 아래와 같이 되게 하시오 !

조인 순서 : dept ----> emp
조인 방법 : 해쉬조인

```
select /*+ leading(d e) use_hash(e) full(d) full(e) */
       e.ename, d.loc
  from emp e, dept d
 where e.deptno = d.deptno(+);
```

경 과: 00:00:00.03

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	574	5 (20)	00:00:01
* 1	HASH JOIN OUTER		14	574	5 (20)	00:00:01
2	TABLE ACCESS FULL	EMP	14	280	2 (0)	00:00:01
3	TABLE ACCESS FULL	DEPT	4	84	2 (0)	00:00:01

0 db block gets
17 consistent gets
0 physical reads

```
select /*+ leading(d e) use_hash(e) full(d) full(e)
       swap_join_inputs(d) */
       e.ename, d.loc
  from emp e, dept d
 where e.deptno = d.deptno(+);
```

경 과: 00:00:00.03

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	574	5 (20)	00:00:01
* 1	HASH JOIN RIGHT OUTER		14	574	5 (20)	00:00:01
2	TABLE ACCESS FULL	DEPT	4	84	2 (0)	00:00:01
3	TABLE ACCESS FULL	EMP	14	280	2 (0)	00:00:01

0 db block gets
7 consistent gets
0 physical reads

문제 59. 아래의 SQL을 튜닝하시오 !

```
select t.calendar_year, sum(s.amount_sold)
  from sales100 s, times100 t
 where s.time_id = t.time_id (+)
       and t.week_ending_day_id = 1581
 group by t.calendar_year;
```

경 과: 00:00:00.09

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		13272	738K	250 (1)	00:00:03
1	HASH GROUP BY		13272	738K	250 (1)	00:00:03
2	NESTED LOOPS					

	3		NESTED LOOPS				13272		738K		249	(0)		00:00:03		3	
	*	4		TABLE ACCESS FULL		TIMES100		7		245		18	(0)		00:00:01		1
	*	5		INDEX RANGE SCAN		SALES100_TIME_ID		1896				2	(0)		00:00:01		2
	6		TABLE ACCESS BY INDEX ROWID		SALES100		1896		41712		115	(0)		00:00:02		4	

0 db block gets
347 consistent gets
0 physical reads

```
create index times100_week_ending_day_id
on times100(week_ending_day_id);
```

```
select t.calendar_year, sum(s.amount_sold)
from sales100 s, times100 t
where s.time_id = t.time_id (+)
and t.week_ending_day_id = 1581
group by t.calendar_year;
```

문제 60. 아래의 SQL을 튜닝하시오! (병렬도 힌트와 FULL힌트를 사용해서 작성하시오 !)

```
create table sales100
as
select * from sh.sales;
```

```
create table times100
as
select * from sh.times;
```

튜닝전 :

```
select t.calendar_year, sum(s.amount_sold)
from sales100 s, times100 t
where s.time_id = t.time_id(+)
and t.week_ending_day_id = 1581
group by t.calendar_year;
```

CALENDAR_YEAR SUM(S.AMOUNT_SOLD)

1998	438660.26
------	-----------

경과: 00:00:01.35

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	
0	SELECT STATEMENT		4	116	1252 (1)	00:00:01	
1	HASH GROUP BY		4	116	1252 (1)	00:00:01	
*	2 HASH JOIN		4386	124K	1251 (1)	00:00:01	
*	3 TABLE ACCESS FULL	TIMES100	7	112	17 (0)	00:00:01	
	4 TABLE ACCESS FULL	SALES100	918K	11M	1231 (1)	00:00:01	

0 db block gets
4492 consistent gets
4485 physical reads

튜닝후 :

```
select /*+ leading(t s) use_hash(s) full(t) full(s) parallel(t 4) parallel(s 4) */
t.calendar_year, sum(s.amount_sold)
```

```

from sales100 s, times100 t
where s.time_id = t.time_id(+)
and t.week_ending_day_id = 1581
group by t.calendar_year;

```

CALENDAR_YEAR SUM(S.AMOUNT_SOLD)

```

-----
1998          438660.26

```

경 과: 00:00:00.40

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
0	SELECT STATEMENT		4	116	348 (1)
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10001	4	116	348 (1)
3	HASH GROUP BY		4	116	348 (1)
4	PX RECEIVE		4	116	348 (1)
5	PX SEND HASH	:TQ10000	4	116	348 (1)
6	HASH GROUP BY		4	116	348 (1)
* 7	HASH JOIN		4386	124K	347 (1)
8	JOIN FILTER CREATE	:BF0000	7	112	5 (0)
* 9	TABLE ACCESS FULL	TIMES100	7	112	5 (0)
10	JOIN FILTER USE	:BF0000	918K	11M	342 (1)
11	PX BLOCK ITERATOR		918K	11M	342 (1)
* 12	TABLE ACCESS FULL	SALES100	918K	11M	342 (1)

Time	TQ	IN-OUT	PQ Distrib
00:00:01			
00:00:01	Q1,01	P->S	QC (RAND)
00:00:01	Q1,01	PCWP	
00:00:01	Q1,01	PCWP	
00:00:01	Q1,00	P->P	HASH
00:00:01	Q1,00	PCWP	
00:00:01	Q1,00	PCWP	
00:00:01	Q1,00	PCWP	
00:00:01	Q1,00	PCWP	
00:00:01	Q1,00	PCWC	
00:00:01	Q1,00	PCWP	

```

0 db block gets
4498 consistent gets
4433 physical reads

```

문제 61. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```

select /*+ leading(s t) use_hash(t) swap_join_inputs(s) */
      t.calendar_year, sum(s.amount_sold)
from sales100 s, times100 t
where s.time_id = t.time_id(+)
group by t.calendar_year;

```

CALENDAR_YEAR SUM(S.AMOUNT_SOLD)

1999	22219947.7
2001	28136462
1998	24083915
2000	23765506.6

경 과: 00:00:00.51

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		5	125		2366 (2)	00:00:01
1	HASH GROUP BY		5	125		2366 (2)	00:00:01
* 2	HASH JOIN OUTER		918K	21M	21M	2344 (1)	00:00:01
3	TABLE ACCESS FULL	SALES100	918K	11M		1231 (1)	00:00:01
4	TABLE ACCESS FULL	TIMES100	1826	21912		17 (0)	00:00:01

0 db block gets
4492 consistent gets
4433 physical reads

튜닝후 1 :

```
select /*+ leading(t s) use_hash(s) swap_join_inputs(t) */
      t.calendar_year, sum(s.amount_sold)
  from sales100 s, times100 t
 where s.time_id = t.time_id(+)
 group by t.calendar_year;
```

CALENDAR_YEAR SUM(S.AMOUNT_SOLD)

1999	22219947.7
2001	28136462
1998	24083915
2000	23765506.6

경 과: 00:00:00.40

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		5	125	1273 (3)	00:00:01
1	HASH GROUP BY		5	125	1273 (3)	00:00:01
* 2	HASH JOIN RIGHT OUTER		918K	21M	1251 (1)	00:00:01
3	TABLE ACCESS FULL	TIMES100	1826	21912	17 (0)	00:00:01
4	TABLE ACCESS FULL	SALES100	918K	11M	1231 (1)	00:00:01

0 db block gets
4492 consistent gets
4433 physical reads

튜닝후 2 :

```
select /*+ leading(t s) use_hash(s) swap_join_inputs(t) full(t) full(s)
      parallel(t 4) parallel(s 4) */
      t.calendar_year, sum(s.amount_sold)
  from sales100 s, times100 t
 where s.time_id = t.time_id(+)
 group by t.calendar_year;
```

CALENDAR_YEAR SUM(S.AMOUNT_SOLD)

1999	22219947.7
2001	28136462

1998 24083915
2000 23765506.6

경 과: 00:00:00.39

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
0	SELECT STATEMENT		5	125	354 (3)
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10001	5	125	354 (3)
3	HASH GROUP BY		5	125	354 (3)
4	PX RECEIVE		5	125	354 (3)
5	PX SEND HASH	:TQ10000	5	125	354 (3)
6	HASH GROUP BY		5	125	354 (3)
* 7	HASH JOIN RIGHT OUTER		918K	21M	347 (1)
8	TABLE ACCESS FULL	TIMES100	1826	21912	5 (0)
9	PX BLOCK ITERATOR		918K	11M	342 (1)
10	TABLE ACCESS FULL	SALES100	918K	11M	342 (1)

Time	TQ	IN-OUT	PQ Distrib
00:00:01			
00:00:01	Q1,01	P->S	QC (RAND)
00:00:01	Q1,01	PCWP	
00:00:01	Q1,01	PCWP	
00:00:01	Q1,00	P->P	HASH
00:00:01	Q1,00	PCWP	
00:00:01	Q1,00	PCWP	
00:00:01	Q1,00	PCWP	
00:00:01	Q1,00	PCWC	
00:00:01	Q1,00	PCWP	

0 db block gets
4498 consistent gets
4433 physical reads

문제 62. 아래의 SQL의 결과를 UNION 으로 구현하시오 !

```
insert into emp( empno, ename, sal, deptno )
values(1929, 'JACK', 4500, 70);
```

```
select e.ename, d.loc
from emp e full outer join dept d
on (e.deptno = d.deptno);
```

ENAME	LOC
JACK	
SMITH	DALLAS
ALLEN	CHICAGO
WARD	CHICAGO
JONES	DALLAS
MARTIN	CHICAGO
BLAKE	CHICAGO
CLARK	NEW YORK

SCOTT DALLAS
 KING NEW YORK
 TURNER CHICAGO
 ADAMS DALLAS
 JAMES CHICAGO
 FORD DALLAS
 MILLER NEW YORK
 BOSTON

경 과: 00:00:00.07

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		15	225	6 (0)	00:00:01
1	VIEW	VW_F0J_0	15	225	6 (0)	00:00:01
* 2	HASH JOIN FULL OUTER		15	300	6 (0)	00:00:01
3	TABLE ACCESS FULL	DEPT	4	44	3 (0)	00:00:01
4	TABLE ACCESS FULL	EMP	14	126	3 (0)	00:00:01

0 db block gets
 14 consistent gets
 0 physical reads

결과 :

```
select e.ename, d.loc
  from emp e, dept d
 where e.deptno = d.deptno(+)
union
select e.ename, d.loc
  from emp e, dept d
 where e.deptno(+) = d.deptno;
```

ENAME	LOC
ADAMS	DALLAS
ALLEN	CHICAGO
BLAKE	CHICAGO
CLARK	NEW YORK
FORD	DALLAS
JACK	
JAMES	CHICAGO
JONES	DALLAS
KING	NEW YORK
MARTIN	CHICAGO
MILLER	NEW YORK
SCOTT	DALLAS
SMITH	DALLAS
TURNER	CHICAGO
WARD	CHICAGO
	BOSTON

경 과: 00:00:00.06

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		29	580	14 (22)	00:00:01
1	SORT UNIQUE		29	580	14 (22)	00:00:01
2	UNION-ALL					
* 3	HASH JOIN OUTER		14	280	6 (0)	00:00:01

4	TABLE ACCESS FULL	EMP	14	126	3 (0)	00:00:01
5	TABLE ACCESS FULL	DEPT	4	44	3 (0)	00:00:01
6	MERGE JOIN OUTER		15	300	6 (17)	00:00:01
7	TABLE ACCESS BY INDEX ROWID	DEPT	4	44	2 (0)	00:00:01
8	INDEX FULL SCAN	PK_DEPT	4		1 (0)	00:00:01
* 9	SORT JOIN		14	126	4 (25)	00:00:01
10	TABLE ACCESS FULL	EMP	14	126	3 (0)	00:00:01

0 db block gets
23 consistent gets
0 physical reads

옛날 버전 방식 :

```
select /*+ optimizer_features_enable('10.2.0.1')
      opt_param('_optimizer_native_full_outer_join','off') */ e.ename, d.loc
from emp e full outer join dept d
on (e.deptno = d.deptno );
```

※ optimizer_features_enable('10.2.0.1') : 옵티마이저를 10g 버전으로 사용하겠다.
opt_param('_optimizer_native_full_outer_join','off') :
full outer join의 성능을 높이는 파라미터를 끄겠다. (옵션 같은것 !)
(예 : 카톡의 알림을 무음으로 !)

ENAME	LOC
MILLER	NEW YORK
KING	NEW YORK
CLARK	NEW YORK
FORD	DALLAS
ADAMS	DALLAS
SCOTT	DALLAS
JONES	DALLAS
SMITH	DALLAS
JAMES	CHICAGO
TURNER	CHICAGO
BLAKE	CHICAGO
MARTIN	CHICAGO
WARD	CHICAGO
ALLEN	CHICAGO
JACK	BOSTON

경 과: 00:00:00.06

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		15	225	13 (16)	00:00:01
1	VIEW		15	225	13 (16)	00:00:01
2	UNION-ALL					
* 3	HASH JOIN OUTER		14	280	7 (15)	00:00:01
4	TABLE ACCESS FULL	EMP	14	126	3 (0)	00:00:01
5	TABLE ACCESS FULL	DEPT	4	44	3 (0)	00:00:01
6	MERGE JOIN ANTI		1	14	6 (17)	00:00:01
7	TABLE ACCESS BY INDEX ROWID	DEPT	4	44	2 (0)	00:00:01
8	INDEX FULL SCAN	PK_DEPT	4		1 (0)	00:00:01
* 9	SORT UNIQUE		14	42	4 (25)	00:00:01
10	TABLE ACCESS FULL	EMP	14	42	3 (0)	00:00:01

0 db block gets
24 consistent gets

0 physical reads

10g 버전에서의 튜닝후:

```
select /*+ opt_param('_optimizer_native_full_outer_join','force') */
       e.ename, d.loc
  from emp e full outer join dept d
    on (e.deptno = d.deptno );
```

※ force : 켜졌다.

문제 63. telecom_price 테이블과 우리반 테이블을 조인해서 학생이름, 나이, 주소, 텔레콤 month_price를 출력하시오 !

```
alter table telecom_price
  add month_price number(20);
```

```
update telecom_price set month_price = 56000 where telecom_name = 'sk';
update telecom_price set month_price = 54000 where telecom_name = 'lg';
update telecom_price set month_price = 52000 where telecom_name = 'kt';
update telecom_price set month_price = 50000 where telecom_name = 'cj hello';
```

```
select e.ename, e.age, e.address, e.telecom, t.month_price
  from emp2 e, telecom_price t
 where lower(e.telecom) = lower(t.telecom_name);
```

경 과 : 00:00:00.15

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	78	5 (20)	00:00:01
* 1	HASH JOIN		1	78	5 (20)	00:00:01
2	TABLE ACCESS FULL	TELECOM_PRICE	4	80	2 (0)	00:00:01
3	TABLE ACCESS FULL	EMP2	28	1624	2 (0)	00:00:01

0 db block gets
7 consistent gets
0 physical reads

문제 64. 위의 결과에서 통신사가 sk인 학생들만 출력하시오 !
(튜닝된 SQL로 작성하시오 !)

```
select /*+ leading(t e) use_nl(e) */
       e.ename, e.age, e.address, e.telecom, t.month_price
  from emp2 e, telecom_price t
 where lower(e.telecom) = lower(t.telecom_name);
       and t.telecom_name = 'sk';
```

경 과 : 00:00:00.10

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	78	5 (0)	00:00:01
1	NESTED LOOPS		1	78	5 (0)	00:00:01
2	TABLE ACCESS FULL	TELECOM_PRICE	4	80	2 (0)	00:00:01
* 3	TABLE ACCESS FULL	EMP2	1	58	1 (0)	00:00:01

0 db block gets

19 consistent gets
0 physical reads

문제 65. 서일 학생의 이름과 나이와 주소와 통신사와 month_price를 출력하는데 튜닝된 SQL로 작성하시오 !
(점심시간 문제)

```
select
    e.ename, e.age, e.address, e.telecom, t.month_price
from emp2 e, telecom_price t
where lower( e.telecom ) = t.telecom_name
    and e.ename = '서일';
```

경 과: 00:00:00.03

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	NESTED LOOPS	
* 3	TABLE ACCESS FULL	EMP2
* 4	INDEX RANGE SCAN	TELECOM_PRICE_TELECOM_NAME
5	TABLE ACCESS BY INDEX ROWID	TELECOM_PRICE

0 db block gets
10 consistent gets
0 physical reads

```
create index telecom_price_telecom_name
on telecom_price(telecom_name);
create index emp2_telecom
on emp2(telecom);

select /*+ leading(e t) use_nl(t) */
    e.ename, e.age, e.address, e.telecom, t.month_price
from emp2 e, telecom_price t
where lower( e.telecom ) = t.telecom_name
    and e.ename = '서일';          /* 1건 */
```

※ 첫 테이블을 액세스하는 순서를 바꿨다, 왜냐하면 emp2 테이블에서 액세스해야 하는 데이터는 서일 데이터 한건 밖에 없기 때문이다

경 과: 00:00:00.01

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	NESTED LOOPS	
* 3	TABLE ACCESS FULL	EMP2
* 4	INDEX RANGE SCAN	TELECOM_PRICE_TELECOM_NAME
5	TABLE ACCESS BY INDEX ROWID	TELECOM_PRICE

0 db block gets
5 consistent gets
0 physical reads

문제 66. 직업이 SALESMAN이고 부서번호가 30번인 사원의 이름과 월급과 직업과 부서위치를 출력하시오 !
(조인 힌트를 사용해서 작성하시오)

```
select e.ename, e.sal, e.job, d.loc
  from emp e, dept d          /* emp 14건, dept 4건*/
 where e.deptno = d.deptno
    and job = 'SALESMAN'      /* 4건 */
    and d.deptno = 30;        /* 1건 */
```

ENAME	SAL	JOB	LOC
MARTIN	1250	SALESMAN	CHICAGO
ALLEN	1600	SALESMAN	CHICAGO
TURNER	1500	SALESMAN	CHICAGO
WARD	1250	SALESMAN	CHICAGO

경 과: 00:00:00.04

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	240	5 (20)	00:00:01
* 1	HASH JOIN		4	240	5 (20)	00:00:01
* 2	TABLE ACCESS FULL	DEPT	1	21	2 (0)	00:00:01
* 3	TABLE ACCESS FULL	EMP	4	156	2 (0)	00:00:01

```
0 db block gets
44 consistent gets
3 physical reads
```

튜닝 후 :

```
create index emp_deptno on emp(deptno);
create index dept_deptno on dept(deptno);
```

```
select /*+ leading(d e) use_nl(e) */
  e.ename, e.sal, e.job, d.loc
  from emp e, dept d
 where e.deptno = d.deptno
    and job = 'SALESMAN'
    and d.deptno = 30;
```

※ 연결고리에서 인덱스가 한쪽이 없으면 없는쪽에서 시작해서 있는 쪽으로 가줘야 한다.

ENAME	SAL	JOB	LOC
MARTIN	1250	SALESMAN	CHICAGO
ALLEN	1600	SALESMAN	CHICAGO
TURNER	1500	SALESMAN	CHICAGO
WARD	1250	SALESMAN	CHICAGO

경 과: 00:00:00.03

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	240	3 (0)	00:00:01
1	NESTED LOOPS					
2	NESTED LOOPS		4	240	3 (0)	00:00:01
* 3	TABLE ACCESS FULL	DEPT	1	21	2 (0)	00:00:01
* 4	INDEX RANGE SCAN	EMP_DEPTNO	5		0 (0)	00:00:01
* 5	TABLE ACCESS BY INDEX ROWID	EMP	4	156	1 (0)	00:00:01

```

0 db block gets
8 consistent gets
0 physical reads

```

```

=====
create table bonus
as
select empno, sal * 1.2 as bonus
from emp;
=====

```

문제 67. 사원이름이 ALLEN인 사원의 이름, 월급, 부서위치, 보너스(bonus)를 출력하시오 !
(조인튜닝 힌트를 사용하시오)

튜닝전 :

```

select e.ename, e.sal, d.loc, b.bonus
from emp e, dept d, bonus b
where e.deptno = d.deptno
and e.empno = b.empno
and e.ename = 'ALLEN';

```

ENAME	SAL LOC	BONUS
ALLEN	1600 CHICAGO	1920

경 과: 00:00:00.01

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	
0	SELECT STATEMENT		1	93	6 (17)	00:00:01	
* 1	HASH JOIN		1	93	6 (17)	00:00:01	
2	NESTED LOOPS						
3	NESTED LOOPS		1	67	3 (0)	00:00:01	
* 4	TABLE ACCESS FULL	EMP	1	46	2 (0)	00:00:01	
* 5	INDEX RANGE SCAN	DEPT_DEPTNO	1		0 (0)	00:00:01	
6	TABLE ACCESS BY INDEX ROWID	DEPT	1	21	1 (0)	00:00:01	
7	TABLE ACCESS FULL	BONUS	14	364	2 (0)	00:00:01	

```

0 db block gets
9 consistent gets
0 physical reads

```

튜닝 후 :

```

alter table emp
add constraint emp_empno_pk primary key(empno);

alter table bonus
add constraint bonus_empno_pk primary key(empno);

select /*+ leading(e d b) use_nl(d) use_nl(b) */
e.ename, e.sal, d.loc, b.bonus
from emp e, dept d, bonus b
where e.deptno = d.deptno
and e.empno = b.empno
and e.ename = 'ALLEN';

```

ENAME	SAL LOC	BONUS
-------	---------	-------

ALLEN

1600 CHICAGO

1920

경 과: 00:00:00.00

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	93	4 (0)	00:00:01
1	NESTED LOOPS					
2	NESTED LOOPS		1	93	4 (0)	00:00:01
3	NESTED LOOPS		1	67	3 (0)	00:00:01
* 4	TABLE ACCESS FULL	EMP	1	46	2 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID	DEPT	1	21	1 (0)	00:00:01
* 6	INDEX RANGE SCAN	DEPT_DEPTNO	1		0 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	BONUS_EMPNO_PK	1		0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	BONUS	1	26	1 (0)	00:00:01

0 db block gets
9 consistent gets
0 physical reads

문제 68. SCOTT과 같은 월급을 받는 사원의 이름과 월급을 출력하시오 !

```
select ename, sal
  from emp
 where sal = (select sal
              from emp
              where ename = 'SCOTT');
```

ENAME	SAL
FORD	3000
SCOTT	3000

경 과: 00:00:00.03

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	20	4 (0)	00:00:01
* 1	TABLE ACCESS FULL	EMP	1	20	2 (0)	00:00:01
* 2	TABLE ACCESS FULL	EMP	1	20	2 (0)	00:00:01

0 db block gets
7 consistent gets
0 physical reads

```
select /*+ qb_name(main) */
       ename, sal
  from emp
 where sal = (select /*+ qb_name(sub) */ sal
              from emp
              where ename = 'SCOTT');
```

Id	Operation	Name	E-Rows	E-Bytes	Cost (%CPU)	E-Time
----	-----------	------	--------	---------	-------------	--------

	0		SELECT STATEMENT						4	(100)			
	*	1		TABLE ACCESS FULL		EMP		1		20		2	(0) 00:00:01
	*	2		TABLE ACCESS FULL		EMP		1		20		2	(0) 00:00:01

Query Block Name / Object Alias (identified by operation id):

1 - MAIN / EMP@MAIN
2 - SUB / EMP@SUB

문제 69. 위의 실행계획이 main query 부터 수행되게 하시오 !

```
select /*+ qb_name(main) */
      ename, sal
  from emp
 where sal = (select /*+ qb_name(sub) no_push_subq */ sal
              from emp
              where ename = 'SCOTT');
```

	Id		Operation		Name		E-Rows		E-Bytes		Cost (%CPU)		E-Time	
	0		SELECT STATEMENT								4	(100)		
	*	1		FILTER										
	2		TABLE ACCESS FULL		EMP		14		280		2	(0)		00:00:01
	*	3		TABLE ACCESS FULL		EMP		1		20		2	(0)	00:00:01

Query Block Name / Object Alias (identified by operation id):

1 - MAIN
2 - MAIN / EMP@MAIN
3 - SUB / EMP@SUB

문제 70. 아래의 SQL을 서브쿼리부터 수행되게도 해보고 메인쿼리부터 수행되게도 해보시오 !

```
select count(*)
  from sales100
 where time_id in ( select time_id
                    from times100
                    where week_ending_day_id = 1581);
```

	Id		Operation		Name	
	0		SELECT STATEMENT			
	1		SORT AGGREGATE			
	2		NESTED LOOPS			
	3		SORT UNIQUE			
	4		TABLE ACCESS BY INDEX ROWID		TIMES100	
	*	5		INDEX RANGE SCAN		TIMES100_WEEK_ENDING_DAY_ID
	*	6		INDEX RANGE SCAN		SALES100_TIME_ID

- 서브쿼리부터 수행

```

select count(*)
  from sales100
 where time_id in ( select /*+ no_unnest push_subq */ time_id
                   from times100
                   where week_ending_day_id = 1581);

COUNT(*)

```

3490

경 과: 00:00:03.20

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	9	1397 (1)	00:00:17
1	SORT AGGREGATE		1	9		
* 2	TABLE ACCESS FULL	SALES100	47399	416K	1379 (1)	00:00:17
* 3	TABLE ACCESS FULL	TIMES100	1	22	18 (0)	00:00:01

0 db block gets
 1952009 consistent gets
 5060 physical reads

- 메인쿼리부터 수행

```

select count(*)
  from sales100
 where time_id in ( select /*+ no_unnest no_pusj_subq */ time_id
                   from times100
                   where week_ending_day_id = 1581);

COUNT(*)

```

3490

경 과: 00:00:03.23

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	9	13M (1)	44:46:00
1	SORT AGGREGATE		1	9		
* 2	FILTER					
3	TABLE ACCESS FULL	SALES100	947K	8331K	1379 (1)	00:00:17
* 4	TABLE ACCESS FULL	TIMES100	1	22	18 (0)	00:00:01

0 db block gets
 1952009 consistent gets
 5060 physical reads

=====

@demobld

=====

문제 71. 아래의 SQL이 순수하게 서브쿼리로 수행되게 하고 서브쿼리부터 실행되게 하시오 !

```
select ename, sal, job
  from emp
 where deptno in ( select deptno from dept) ;
```

경 과: 00:00:00.05

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	728	5 (20)	00:00:01
* 1	HASH JOIN SEMI		14	728	5 (20)	00:00:01
2	TABLE ACCESS FULL	EMP	14	546	2 (0)	00:00:01
3	TABLE ACCESS FULL	DEPT	4	52	2 (0)	00:00:01

0 db block gets
7 consistent gets
0 physical reads

```
select ename, sal, job
  from emp
 where deptno in ( select /*+ no_unnest push_subq */deptno from dept) ;
```

경 과: 00:00:00.04

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	39	4 (0)	00:00:01
* 1	TABLE ACCESS FULL	EMP	1	39	2 (0)	00:00:01
* 2	TABLE ACCESS FULL	DEPT	1	13	2 (0)	00:00:01

0 db block gets
21 consistent gets
0 physical reads

문제 72. 위의 SQL이 메인쿼리부터 수행되게 하시오 !

```
select ename, sal, job
  from emp
 where deptno in (select /*+ no_unnest no_push_subq */ deptno from dept);
```

경 과: 00:00:00.04

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	546	4 (0)	00:00:01
* 1	FILTER					
2	TABLE ACCESS FULL	EMP	14	546	2 (0)	00:00:01
* 3	TABLE ACCESS FULL	DEPT	1	13	2 (0)	00:00:01

0 db block gets
13 consistent gets
0 physical reads

문제 73. 안혜진 학생과 같은 전공인 학생들의 이름과 전공을 출력하시오 !
(실행계획을 서브쿼리부터 수행되게 하시오 !)

```
select ename, major
  from emp2
 where major = (select /*+ no_unnest push_subq */ major from emp2 where ename = '안혜진');
```

경 과: 00:00:00.02

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	24	4 (0)	00:00:01
* 1	TABLE ACCESS FULL	EMP2	1	24	2 (0)	00:00:01
* 2	TABLE ACCESS FULL	EMP2	1	24	2 (0)	00:00:01

0 db block gets
5 consistent gets
0 physical reads

문제 74. 아래의 SQL이 조인으로 풀리게 힌트를 주고 조인방법중에 nested loop join이 되게 하시오 !

```
select ename, sal, job
  from emp
 where deptno in ( select deptno from dept );
```

```
select ename, sal, job
  from emp
 where deptno in ( select /*+ unnest nl_sj */deptno from dept );
```

경 과: 00:00:00.04

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	728	8 (0)	00:00:01
1	NESTED LOOPS SEMI		14	728	8 (0)	00:00:01
2	TABLE ACCESS FULL	EMP	14	546	2 (0)	00:00:01
* 3	TABLE ACCESS FULL	DEPT	4	52	0 (0)	00:00:01

0 db block gets
13 consistent gets
0 physical reads

```
select ename, sal, job
  from emp
 where deptno in ( select /*+ unnest hash_sj */deptno from dept );
```

경 과: 00:00:00.04

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	728	5 (20)	00:00:01
* 1	HASH JOIN SEMI		14	728	5 (20)	00:00:01
2	TABLE ACCESS FULL	EMP	14	546	2 (0)	00:00:01
3	TABLE ACCESS FULL	DEPT	4	52	2 (0)	00:00:01

```

0 db block gets
15 consistent gets
0 physical reads

```

문제 75. 위의 해쉬세미조인의 조인순서를 dept ----> emp 로 변경하시오 !

```

select ename, sal, job
  from emp
 where deptno in ( select /*+ unnest hash_sj swap_join_inputs(d) */deptno from dept d);

```

경 과: 00:00:00.04

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	728	5 (20)	00:00:01
* 1	HASH JOIN RIGHT SEMI		14	728	5 (20)	00:00:01
2	TABLE ACCESS FULL	DEPT	4	52	2 (0)	00:00:01
3	TABLE ACCESS FULL	EMP	14	546	2 (0)	00:00:01

```

0 db block gets
15 consistent gets
0 physical reads

```

문제 76. 아래의 SQL을 hash join right semi 조인이 되게 하시오 !

튜닝전 :

```

select count(*)
  from sales100
 where time_id in ( select /*+ no_unnest no_push_subq */ time_id
                    from times100
                    where week_ending_day_id = 1581 );

```

경 과: 00:00:03.26

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	9	13M (1)	44:46:00
1	SORT AGGREGATE		1	9		
* 2	FILTER					
3	TABLE ACCESS FULL	SALES100	947K	8331K	1379 (1)	00:00:17
* 4	TABLE ACCESS FULL	TIMES100	1	22	18 (0)	00:00:01

```

0 db block gets
1952009 consistent gets
5060 physical reads

```

```

select count(*)
  from sales100
 where time_id in ( select /*+ unnest hash_sj swap_join_inputs(times100) */ time_id
                    from times100
                    where week_ending_day_id = 1581 );

```

경 과: 00:00:00.22

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	31	1400 (1)	00:00:17

	1		SORT AGGREGATE				1		31					
	*	2		HASH JOIN RIGHT SEMI				13272		401K		1400		(1) 00:00:17
	*	3		TABLE ACCESS FULL		TIMES100		7		154		18		(0) 00:00:01
		4		TABLE ACCESS FULL		SALES100		947K		8331K		1379		(1) 00:00:17

0 db block gets
5127 consistent gets
5060 physical reads

문제 77. 아래와 같이 실행계획이 나오게 하시오 !

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time	
0	SELECT STATEMENT		1	31		2343 (1)	00:00:29	
1	SORT AGGREGATE		1	31				
* 2	HASH JOIN SEMI		13272	401K	18M	2343 (1)	00:00:29	
3	TABLE ACCESS FULL	SALES100	947K	8331K		1379 (1)	00:00:17	
* 4	TABLE ACCESS FULL	TIMES100	7	154		18 (0)	00:00:01	

```
select /*+ leading(s t) */ count(*)
  from sales100 s
 where time_id in ( select /*+ unnest hash_sj swap_join_inputs(s) */ time_id
                   from times100 t
                   where week_ending_day_id = 1581 );
```

문제 78. 관리자가 아닌 직원들의 이름을 출력하시오 !
(자기 밑에 직속부하가 한명도 없는 직원들)

```
select ename
  from emp
 where empno not in ( select nvl(mgr,0) from emp);
```

```
select ename
  from emp
 where empno not in ( select /*+ unnest hash_aj */ nvl(mgr,0) from emp);
```

경 과: 00:00:00.03

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	
0	SELECT STATEMENT		14	462	5 (20)	00:00:01	
* 1	HASH JOIN ANTI		14	462	5 (20)	00:00:01	
2	TABLE ACCESS FULL	EMP	14	280	2 (0)	00:00:01	
3	TABLE ACCESS FULL	EMP	14	182	2 (0)	00:00:01	

0 db block gets
14 consistent gets
0 physical reads

문제 79. 아래의 SQL을 해쉬 안티 조인이 되게 하시오 !

```
select *
  from emp
 where deptno not in (select deptno from dept);
```

경 과: 00:00:00.01

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	1400	5 (20)	00:00:01
* 1	HASH JOIN ANTI NA		14	1400	5 (20)	00:00:01
2	TABLE ACCESS FULL	EMP	14	1218	2 (0)	00:00:01
3	TABLE ACCESS FULL	DEPT	4	52	2 (0)	00:00:01

0 db block gets
14 consistent gets
0 physical reads

문제 80. 위의 SQL의 실행계획이 아래와 같이 되게 하시오 !

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	1400	5 (20)	00:00:01
* 1	HASH JOIN RIGHT ANTI		14	1400	5 (20)	00:00:01
* 2	TABLE ACCESS FULL	DEPT	4	52	2 (0)	00:00:01
* 3	TABLE ACCESS FULL	EMP	14	1218	2 (0)	00:00:01

```
select /*+ leading(d e) */ *
  from emp e
 where deptno not in (select /*+ unnest hash_aj swap_join_inputs(d) */ deptno
                      from dept d
                      where deptno is not null)
 and deptno is not null;
```

경 과: 00:00:00.01

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	1400	5 (20)	00:00:01
* 1	HASH JOIN RIGHT ANTI		14	1400	5 (20)	00:00:01
* 2	TABLE ACCESS FULL	DEPT	4	52	2 (0)	00:00:01
* 3	TABLE ACCESS FULL	EMP	14	1218	2 (0)	00:00:01

0 db block gets
14 consistent gets
0 physical reads

문제 81. 아래의 SQL을 분석함수를 이용하지 않은 SQL로 작성하시오 !
(오늘의 마지막 문제)

전 :

```
select empno, ename, sal, sum(sal) over (order by empno) 누적치
from emp;
```

후 :

```
with emp_sal as (select rownum as rn, e.*
                  from (select empno, ename, sal
                        from emp
                        order by empno) e)
select e1.empno, e1.ename, e1.sal,
       (select sum(sal)
        from emp_sal e2
        where e2.rn <= e1.rn) 누적치
from emp_sal e1;

select empno, ename, sal, (select sum(sal)
                           from emp ee
                           where ee.empno <= e.empno) 누적치
from emp e
order by empno;
```

문제 82. 아래의 수동 파티셔닝을 파티션 뷰로 구현하시오 !

```
=====
@demobld
=====
```

1. 파티션 뷰를 정의할 때 사용할 base 테이블을 만든다.

```
create table p1 as select * from emp where deptno = 10;
create table p2 as select * from emp where deptno = 20;
create table p3 as select * from emp where deptno = 30;
```

2. 체크제약을 반드시 설정해야 함

```
alter table p1 add constraint c_deptno_10 check(deptno <20);
alter table p2 add constraint c_deptno_20 check(deptno>=20 and deptno<30);
alter table p3 add constraint c_deptno_30 check(deptno>=30);
```

3. 인덱스를 생성한다.

```
create index p1_empno_idx on p1(empno);
create index p2_empno_idx on p2(empno);
create index p3_empno_idx on p3(empno);
```

4. 3개의 테이블에 대하여 분석작업을 한다

```
analyze table p1 compute statistics;
analyze table p2 compute statistics;
analyze table p3 compute statistics;
```

문제 83. p2 테이블에 아래의 data를 입력하시오 !

```
empno : 3829
ename : jack
sal   : 4000
deptno : 30
```

```
insert into p2(empno, ename, sal, deptno)
values (3829, 'jack', 4000, 30);
```

ORA-02290: check constraint (HEAVEN.C_DEPTNO_20) violated

※ 설명 : p1 테이블에는 부서번호 10번만 입력할 수 있고
p2 테이블에는 부서번호 20번만 입력할 수 있고
p3 테이블에는 부서번호 30번만 입력할 수 있다.

문제 84. 위에서 준비된 3개의 테이블(segment)을 이용해서 파티션 view를 생성하시오 !

```
create or replace view emp_partition
as
select * from p1
union all
select * from p2
union all
select * from p3;

select * from emp_partition;
```

문제 85. 부서번호 20번인 직원들의 모든 컬럼을 출력하는 아래의 2개의 SQL의 성능을 비교해 보시오 !
(block의 갯수)

```
1. select *
   from emp_partition
   where deptno = 20;
```

경 과: 00:00:00.03

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		5	170	3 (0)	00:00:01
1	VIEW	EMP_PARTITION	5	170	3 (0)	00:00:01
2	UNION-ALL					
* 3	FILTER					
* 4	TABLE ACCESS FULL	P1	1	31	2 (0)	00:00:01
* 5	TABLE ACCESS FULL	P2	5	155	2 (0)	00:00:01
* 6	FILTER					
* 7	TABLE ACCESS FULL	P3	1	34	2 (0)	00:00:01

0 db block gets
3 consistent gets
0 physical reads

```
2. select *
   from emp
   where deptno = 20;
```

경 과: 00:00:00.03

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		5	435	2 (0)	00:00:01
* 1	TABLE ACCESS FULL	EMP	5	435	2 (0)	00:00:01

0 db block gets
4 consistent gets
0 physical reads

문제 86. 이번에는 파티션 뷰 말고 실제로 파티션 테이블을 생성하시오 !

```
create table emp_partition2
partition by range(deptno)                /* 부서번호별로 파티션 하겠다 */
(
    partition p1 values less than (20),    /* 부서번호 10번의 데이터를 구성하겠다. */
    partition p2 values less than (30),    /* 부서번호 20번의 데이터를 구성하겠다. */
    partition p3 values less than (40)     /* 부서번호 30번의 데이터를 구성하겠다. */
)
as
select * from emp;                        /* 파티션 하고 싶은 테이블 */

select * from emp_partition2;
```

* 아래의 2개의 SQL의 block의 갯수의 차이가 있는지 확인하시오 !

```
select * from emp_partition2 where deptno = 20;
```

경 과: 00:00:00.03

Id	Operation	Name
0	SELECT STATEMENT	
1	PARTITION RANGE SINGLE	
* 2	TABLE ACCESS FULL	EMP_PARTITION2

0 db block gets
4 consistent gets
0 physical reads

```
select * from emp where deptno = 20;
```

경 과: 00:00:00.03

Id	Operation	Name
0	SELECT STATEMENT	
* 1	TABLE ACCESS FULL	EMP

0 db block gets
8 consistent gets
0 physical reads

문제 87. 우리반 테이블을 12c database에 구현하시오 !

12c database(orcl) -----> 11g database(xe)

```
create public database link link_11g
connect to heaven
identified by heaven
using 'localhost:1521/xe';
```

```
select * from tab@link_11g;
```

```
create table emp2
as
select * from emp2@link_11g;
```

문제 88. 우리반 테이블로 range 파티션 테이블을 생성하시오 !

```
create table emp2_partition
partition by range(age)
(
    partition p1 values less than (27),
    partition p2 values less than (29),
    partition p3 values less than (31),
    partition p4 values less than (45)
)
as
select * from emp2;
```

문제 89. 나이가 26살인 학생들의 이름과 나이를 출력하는데 하나는 파티션테이블에서 조회하고 하나는 우리반 테이블에서 조회해서 성능상의 차이가 생기는지 확인하시오 !

1. 파티션 테이블

```
select ename, age
from emp2_partition
where age = 26;
```

경 과: 00:00:00.05

Id	Operation	Name
0	SELECT STATEMENT	
1	PARTITION RANGE SINGLE	
* 2	TABLE ACCESS FULL	EMP2_PARTITION

```
0 db block gets
4 consistent gets
0 physical reads
```

2. 우리반테이블

```
select ename, age
from emp2
where age = 26;
```

경 과: 00:00:00.04

Id	Operation	Name
----	-----------	------

```

| 0 | SELECT STATEMENT |      |
|* 1 | TABLE ACCESS FULL| EMP2 |
-----
0 db block gets
4 consistent gets
0 physical reads

```

문제 90. 우리반 테이블을 해쉬 파티션으로 생성하시오 (파티션 테이블 이름 : emp2_hash_partition)

```

create table emp2_hash_partition
partition by hash(age) partitions 3
as
select * from emp2;

```

```

exec dbms_stats.gather_table_stats('SCOTT', 'EMP2_HASH_PARTITION');

```

```

select table_name, partition_name, num_rows
from user_tab_partitions
where table_name = 'EMP2_HASH_PARTITION';

```

TABLE_NAME	PARTITION_NAME	NUM_ROWS
EMP2_HASH_PARTITION	SYS_P289	9
EMP2_HASH_PARTITION	SYS_P290	15
EMP2_HASH_PARTITION	SYS_P291	3

문제 91. 사원 테이블의 부서번호로 list 파티션을 생성하시오 !

```

create table emp_list_partition
partition by list(deptno)
(
partition p1 values ('10'),
partition p2 values ('20'),
partition p3 values ('30')
)
as
select * from emp;

```

```

exec dbms_stats.gather_table_stats('SCOTT', 'EMP_LIST_PARTITION');

```

```

select table_name, partition_name, num_rows
from user_tab_partitions
where table_name = 'EMP_LIST_PARTITION';

```

TABLE_NAME	PARTITION_NAME	NUM_ROWS
EMP_LIST_PARTITION	P1	3
EMP_LIST_PARTITION	P2	5
EMP_LIST_PARTITION	P3	6

문제 92. 우리반 테이블을 통신사별로 나눠서 리스트 파티션 테이블을 생성하시오 ! (점심시간 문제)
(골고루 데이터가 분배되었는지 보여주세요 !)

```
create table emp2_list_partition
partition by list(telecom)
(
    partition p1 values ('sk'),
    partition p2 values ('lg'),
    partition p3 values ('kt')
)
as
select * from emp2 e;

exec dbms_stats.gather_table_stats('SCOTT', 'EMP2_LIST_PARTITION');

select table_name, partition_name, num_rows
      from user_tab_partitions
     where table_name = 'EMP2_LIST_PARTITION';
```

TABLE_NAME	PARTITION_NAME	NUM_ROWS
EMP2_LIST_PARTITION	P1	13
EMP2_LIST_PARTITION	P2	5
EMP2_LIST_PARTITION	P3	9

문제 93. 점심시간 문제로 만들었던 통신사별로 구분한 파티션 테이블의 쿼리문을 작성해서 파티션 프루닝을 하는지 확인하시오 !

```
select *
  from emp2_list_partition
 where telecom = 'sk';
```

경 과: 00:00:00.09

Id	Operation	Name
0	SELECT STATEMENT	
1	PARTITION LIST SINGLE	
2	TABLE ACCESS FULL	EMP2_LIST_PARTITION

```
0 db block gets
4 consistent gets
0 physical reads
```

```
=====
drop view emp_partition;
drop table emp_partition;
=====
```


문제 94. emp_partition 파티션 테이블과 dept_partition 파티션 테이블을 deptno를 파티션 키로 해서 생성하시오 !

```
create table emp_partition
partition by range(deptno)
(
    partition p1 values less than (20),
    partition p2 values less than (30),
    partition p3 values less than (40),
    partition p4 values less than (50)
)
as
select * from emp;

create table dept_partition
partition by range(deptno)
(
    partition p1 values less than (20),
    partition p2 values less than (30),
    partition p3 values less than (40),
    partition p4 values less than (50)
)
as
select * from dept;

select * from emp_partition;
select * from dept_partition;
```

문제 95. emp_partition과 dept_partition을 조인해서 이름과 부서위치와 부서번호를 출력하고 실행계획을 확인하시오 !

```
select /* gather_plan_statistics */ e.ename, d.loc, e.deptno
from emp_partition e, dept_partition d
where e.deptno = d.deptno;
```

Id	Operation	Name	E-Rows
0	SELECT STATEMENT		
1	NESTED LOOPS		14
2	PARTITION RANGE ALL		4
3	TABLE ACCESS FULL	DEPT_PARTITION	4
4	PARTITION RANGE ITERATOR		4
* 5	TABLE ACCESS FULL	EMP_PARTITION	4

※ 해쉬조인으로 유도를 해야 파티션 와이즈조인을 한다.

문제 96. 위의 조인문을 해쉬조인으로 유도해서 파티션 와이즈 조인이 되게 하시오 !

```
select /*+ leading(d e) use_hash(e) full(d) full(e)
parallel(e 2) parallel(d 2)
pq_distribute(e,none,none) */
e.ename, d.loc, e.deptno
from emp_partition e, dept_partition d
where e.deptno = d.deptno;
```

Id	Operation	Name
----	-----------	------

	0		SELECT STATEMENT			
	1		PX COORDINATOR			
	2		PX SEND QC (RANDOM)		:TQ10000	
	3		PX PARTITION RANGE ALL			
	* 4		HASH JOIN			
	5		TABLE ACCESS FULL		DEPT_PARTITION	
	6		TABLE ACCESS FULL		EMP_PARTITION	

```
=====
create table telecom_price
as
select * from telecom_price@link_11g;
=====
```

문제 97. 우리반 (emp2) 테이블과 telecom_price테이블을 조인해서 이름과 나이와 텔레콤과 month_price를 출력하는데 파티션 와이즈 조인이 되게 하시오 !

```
select/*+ gather_plan_statistics
      leading(t e) use_hash(e) full(t) full(e)
      parallel(t 2) parallel(e 2)
      pq_distribute(e, hash, hash) */
      e.ename, e.age, e.telecom, t.month_price
from emp2 e, telecom_price t
where e.telecom = t.telecom_name;
```

	Id		Operation		Name	
	0		SELECT STATEMENT			
	1		PX COORDINATOR			
	2		PX SEND QC (RANDOM)		:TQ10002	
	* 3		HASH JOIN BUFFERED			
	4		PX RECEIVE			
	5		PX SEND HYBRID HASH		:TQ10000	
	6		STATISTICS COLLECTOR			
	7		PX BLOCK ITERATOR			
	* 8		TABLE ACCESS FULL		TELECOM_PRICE	
	9		PX RECEIVE			
	10		PX SEND HYBRID HASH		:TQ10001	
	11		PX BLOCK ITERATOR			
	* 12		TABLE ACCESS FULL		EMP2	

문제 98. emp_partition 테이블의 deptno에 로컬 파티션 인덱스를 생성하시오 !

```
create index emp_partition_local
on emp_partition(deptno) local;
```

```
select bytes, blocks
from user_segments
where segment_name = 'EMP_PARTITION_LOCAL';
```

BYTES	BLOCKS
8454144	1032
8454144	1032
8454144	1032

문제 99. sales 테이블의 데이터를 sales 테이블의 구조만 가지고 있는 sales 500 테이블에 입력할 때 병렬로 입력했을 때와 병렬이 아닌 serial로 입력했을 때와 속도 차이를 확인하시오 !

create table sales		set timing on
as		
select * from sh.sales;		insert into sales500
		select * from sales;
select count(*) from sales;		
		918843 행이 생성되었습니다.
create table sales500		경 과: 00:00:02.26
as		
select * from sales		truncate table sales500;
where 1=2;		
		alter session enable parallel dml;
		insert /* parallel(s5 4) */ into sales500 s5
		select * from sales;
		918843 행이 생성되었습니다.
		경 과: 00:00:01.18

문제 100. 이름과 부서위치와 월급을 출력하는 view를 생성하시오 !
(view 이름은 emp100으로 하세요)

```
create view emp100
as
select e.ename, d.loc, e.sal
from emp e, dept d
where e.deptno = d.deptno;
```

```
select * from emp100;
```

문제 101. emp100과 salgrade를 조인해서 이름과 월급과 부서위치와 급여등급을 출력하시오 !

```
select e.ename, e.sal, e.loc, s.grade
from emp100 e, salgrade s
where e.sal between s.losal and s.hisal;
```

↓ Query transformer가 view를 해체하고 ~

```
select e.ename, e.sal, d.loc, s.grade
from emp e, dept d, salgrade s
where e.deptno = d.deptno
and e.sal between s.losal and s.hisal;
```

	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
	0	SELECT STATEMENT		1	93	8 (13)	00:00:01
*	1	HASH JOIN		1	93	8 (13)	00:00:01
	2	MERGE JOIN CARTESIAN		20	1200	5 (0)	00:00:01
	3	TABLE ACCESS FULL	DEPT	4	84	2 (0)	00:00:01
	4	BUFFER SORT		5	195	3 (0)	00:00:01
	5	TABLE ACCESS FULL	SALGRADE	5	195	1 (0)	00:00:01
	6	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

※ 실행계획에 emp100이 없다 ??? 로지컬 옵티마이저가 emp100 뷰를 알아서 해체 했다!

지금 상태의 최선의 튜닝

```
select /*+ leading(s e) use_nl(e) */
      e.ename, e.sal, e.loc, s.grade
from emp100 e, salgrade s
where e.sal between s.losal and s.hisal;
```

	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
	0	SELECT STATEMENT		1	93	8 (13)	00:00:01
*	1	HASH JOIN		1	93	8 (13)	00:00:01
	2	MERGE JOIN CARTESIAN		20	1200	5 (0)	00:00:01
	3	TABLE ACCESS FULL	DEPT	4	84	2 (0)	00:00:01
	4	BUFFER SORT		5	195	3 (0)	00:00:01
	5	TABLE ACCESS FULL	SALGRADE	5	195	1 (0)	00:00:01
	6	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

※ 옵티마이저가 SQL을 해체해버리기 때문에 힌트가 무용지물이 되어 버린다.

문제 102. 아래의 SQL의 힌트가 작동되도록 새로운 힌트를 주시오 !

```
select /*+ leading(s e) use_nl(e) */
      e.ename, e.sal, e.loc, s.grade
from emp100 e, salgrade s
where e.sal between s.losal and s.hisal;
```

↓

```
select /*+ no_merge(e1) leading(s e1) use_nl(e1) */
      e1.ename, e1.sal, e1.loc, s.grade
from emp100 e1, salgrade s
where e1.sal between s.losal and s.hisal;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	67	25 (12)	00:00:01
1	NESTED LOOPS		1	67	25 (12)	00:00:01
2	TABLE ACCESS FULL	SALGRADE	5	195	2 (0)	00:00:01
* 3	VIEW	EMP100	1	28	5 (20)	00:00:01
* 4	HASH JOIN		14	756	5 (20)	00:00:01
5	TABLE ACCESS FULL	DEPT	4	84	2 (0)	00:00:01
6	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

※ no_merge를 썼더니 뷰가 해체되지 않고 원하는 실행계획이 나왔다.

문제 103. 위의 실행계획에서 emp100안의 emp와 dept의 조인순서를 변경하시오 !

```
select /*+ no_merge(e1) leading(s e1) use_nl(e1)
      leading(e1.e e1.d) use_nl(e1.d) */
      e1.ename, e1.sal, e1.loc, s.grade
from emp100 e1, salgrade s
where e1.sal between s.losal and s.hisal;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	67	42 (0)	00:00:01
1	NESTED LOOPS		1	67	42 (0)	00:00:01
2	TABLE ACCESS FULL	SALGRADE	5	195	2 (0)	00:00:01
* 3	VIEW	EMP100	1	28	8 (0)	00:00:01
4	NESTED LOOPS		14	756	8 (0)	00:00:01
5	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01
* 6	TABLE ACCESS FULL	DEPT	1	21	0 (0)	00:00:01

※ 뷰는 생성시 쿼리를 날려 테이블처럼 보여주는 건데 안의 쿼리를 수정하고싶다면 e1.[뷰 속의 테이블 별칭]을 이용해서 수정이 가능하다.
그래서 처음에 쿼리에 e를 쓰지않고 이것을 위해 e1으로 수정한것 !

문제 104. 위의 SQL로 아래의 실행계획이 나오게 하시오 !

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	67	9 (34)	00:00:01
1	MERGE JOIN		1	67	9 (34)	00:00:01
2	SORT JOIN		14	392	6 (34)	00:00:01
3	VIEW	EMP100	14	392	5 (20)	00:00:01
* 4	HASH JOIN		14	756	5 (20)	00:00:01
5	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01
6	TABLE ACCESS FULL	DEPT	4	84	2 (0)	00:00:01
* 7	FILTER					
* 8	SORT JOIN		5	195	3 (34)	00:00:01
9	TABLE ACCESS FULL	SALGRADE	5	195	2 (0)	00:00:01

```
select /*+ no_merge(e1) leading(e1 s) use_merge(s)
      leading(e1.d e1.e) use_hash(e1.e) full(e1.d) full(e1.e) */
      e1.ename, e1.sal, e1.loc, s.grade
from emp100 e1, salgrade s
where e1.sal between s.losal and s.hisal;
```

문제 105. 위의 SQL의 emp100을 in line view로 풀어서 작성하시오 !

```
select e1.ename, e1.sal, e1.loc, s.grade
  from (select e.ename, e.sal, d.loc
        from emp e, dept d
        where e.deptno = d.deptno) e1, salgrade s
 where e1.sal between s.losal and s.hisal;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	93	8 (13)	00:00:01
* 1	HASH JOIN		1	93	8 (13)	00:00:01
2	MERGE JOIN CARTESIAN		20	1200	5 (0)	00:00:01
3	TABLE ACCESS FULL	DEPT	4	84	2 (0)	00:00:01
4	BUFFER SORT		5	195	3 (0)	00:00:01
5	TABLE ACCESS FULL	SALGRADE	5	195	1 (0)	00:00:01
6	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

문제 106. 위의 SQL의 in line view를 해체하지 못하도록 힌트를 작성하시오 !

첫번째 방법

```
select /*+ no_merge(e1) */ e1.ename, e1.sal, e1.loc, s.grade
  from (select e.ename, e.sal, d.loc
        from emp e, dept d
        where e.deptno = d.deptno) e1, salgrade s
 where e1.sal between s.losal and s.hisal;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	67	9 (34)	00:00:01
1	MERGE JOIN		1	67	9 (34)	00:00:01
2	SORT JOIN		5	195	3 (34)	00:00:01
3	TABLE ACCESS FULL	SALGRADE	5	195	2 (0)	00:00:01
* 4	FILTER					
* 5	SORT JOIN		14	392	6 (34)	00:00:01
6	VIEW		14	392	5 (20)	00:00:01
* 7	HASH JOIN		14	756	5 (20)	00:00:01
8	TABLE ACCESS FULL	DEPT	4	84	2 (0)	00:00:01
9	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

두번째 방법

```
select e1.ename, e1.sal, e1.loc, s.grade
  from (select /*+ no_merge */ e.ename, e.sal, d.loc
        from emp e, dept d
        where e.deptno = d.deptno) e1, salgrade s
 where e1.sal between s.losal and s.hisal;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	67	9 (34)	00:00:01
1	MERGE JOIN		1	67	9 (34)	00:00:01
2	SORT JOIN		5	195	3 (34)	00:00:01
3	TABLE ACCESS FULL	SALGRADE	5	195	2 (0)	00:00:01
* 4	FILTER					
* 5	SORT JOIN		14	392	6 (34)	00:00:01
6	VIEW		14	392	5 (20)	00:00:01

	*	7		HASH JOIN				14		756		5	(20)		00:00:01	
		8		TABLE ACCESS FULL		DEPT		4		84		2	(0)		00:00:01	
		9		TABLE ACCESS FULL		EMP		14		462		2	(0)		00:00:01	

```
=====
@demobld
```

```
create index emp_ename on emp(ename);
=====
```

문제 107. 이름에 EN 또는 IN을 포함하고 있는 사원들의 이름과 월급과 직업과 부서번호를 출력하시오 !

튜닝 전 :

```
select ename, sal, job, deptno
  from emp
 where ename like '%EN%'
        or ename like '%IN%';
```

	Id		Operation		Name		Rows		Bytes		Cost (%CPU)		Time	
	0		SELECT STATEMENT				3		117		2 (0)		00:00:01	
	*		TABLE ACCESS FULL		EMP		3		117		2 (0)		00:00:01	

튜닝 후(1) :

```
select e.ename, e.sal, e.job, e.deptno
  from emp e, (select /*+ index_ffs(emp emp_ename) */ rowid rn
                from emp
                where ename like '%EN%'
                       or ename like '%IN%') v
 where e.rowid = v.rn;
```

	Id		Operation		Name		Rows		Bytes		Cost (%CPU)		Time	
	0		SELECT STATEMENT				3		153		2 (0)		00:00:01	
	*		TABLE ACCESS FULL		EMP		3		153		2 (0)		00:00:01	

※ in line view를 자꾸 해체해버려서 힌트가 소용없다,,

튜닝 후(2) :

```
select e.ename, e.sal, e.job, e.deptno
  from emp e, (select /*+ no_merge index_ffs(emp emp_ename) */ rowid rn
                from emp
                where ename like '%EN%'
                       or ename like '%IN%') v
 where e.rowid = v.rn;
```

	Id		Operation		Name		Rows		Bytes		Cost (%CPU)		Time	
	0		SELECT STATEMENT				3		189		5 (20)		00:00:01	
	*		HASH JOIN				3		189		5 (20)		00:00:01	
	2		VIEW				3		36		2 (0)		00:00:01	
	*		TABLE ACCESS FULL		EMP		3		57		2 (0)		00:00:01	
	4		TABLE ACCESS FULL		EMP		14		714		2 (0)		00:00:01	

```

0 db block gets
7 consistent gets
0 physical reads
    ※ in line view 해체하지 못하게 힌트를 썼다

```

튜닝 후(3) :

```

select /*+ leading(v e) use_nl(e) */ e.ename, e.sal, e.job, e.deptno
  from emp e, (select /*+ no_merge index_ffs(emp emp_ename) */ rowid rn
                from emp
                where ename like '%EN%'
                or ename like '%IN%') v
 where e.rowid = v.rn;

```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	189	5 (0)	00:00:01
1	NESTED LOOPS		3	189	5 (0)	00:00:01
2	VIEW		3	36	2 (0)	00:00:01
* 3	TABLE ACCESS FULL	EMP	3	57	2 (0)	00:00:01
4	TABLE ACCESS BY USER ROWID	EMP	1	51	1 (0)	00:00:01

```

0 db block gets
6 consistent gets
0 physical reads
    ※ 테이블이 작기 때문에 nested loop join을 수행하게 힌트를 썼다.

```

문제 108. 사원번호, 이름, 월급, 사원테이블의 최대월급,
 사원테이블의 최소월급,
 사원테이블의 토달월급,
 사원테이블의 평균월급을 출력하시오 !

튜닝 전 :

```

select empno, ename, sal,
       (select max(sal) from emp) 최대월급,
       (select min(sal) from emp) 최소월급,
       (select sum(sal) from emp) 토달월급,
       (select round(avg(sal)) from emp) 평균월급
  from emp;

```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	462	2 (0)	00:00:01
1	SORT AGGREGATE		1	13		
2	TABLE ACCESS FULL	EMP	14	182	2 (0)	00:00:01
3	SORT AGGREGATE		1	13		
4	TABLE ACCESS FULL	EMP	14	182	2 (0)	00:00:01
5	SORT AGGREGATE		1	13		
6	TABLE ACCESS FULL	EMP	14	182	2 (0)	00:00:01
7	SORT AGGREGATE		1	13		
8	TABLE ACCESS FULL	EMP	14	182	2 (0)	00:00:01
9	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

```

0 db block gets
16 consistent gets
0 physical reads

```

튜닝 후(내꺼) :

```

select e.empno, e.ename, e.sal, s.mx, s.mn, s.su, s.ag

```



```
from emp e, (select max(sal) mx, min(sal) mn, sum(sal) su, round(avg(sal)) ag
from emp) s;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	1190	4 (0)	00:00:01
1	NESTED LOOPS		14	1190	4 (0)	00:00:01
2	VIEW		1	52	2 (0)	00:00:01
3	SORT AGGREGATE		1	13		
4	TABLE ACCESS FULL	EMP	14	182	2 (0)	00:00:01
5	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

```
0 db block gets
7 consistent gets
0 physical reads
```

튜닝 후(선생님) :

```
select empno, ename, sal, substr(total,1,10 ) 최대,
      substr(total,11,10 ) 최소,
      substr(total,21,10 ) 토탈,
      substr(total,31,10 ) 평균
from (select empno, ename, sal,
      (select rpad(max(sal),10,' '))||
      rpad(min(sal),10,' ')||
      rpad(sum(sal),10,' ')||
      rpad(round(avg(sal)),10,' ')
      from emp) total
from emp);
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	462	2 (0)	00:00:01
1	SORT AGGREGATE		1	13		
2	TABLE ACCESS FULL	EMP	14	182	2 (0)	00:00:01
3	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

```
0 db block gets
7 consistent gets
0 physical reads
```

튜닝 후(배운거 활용) :

```
select empno, ename, sal, substr(total,1,10 ) 최대,
      substr(total,11,10 ) 최소,
      substr(total,21,10 ) 토탈,
      substr(total,31,10 ) 평균
from (select /*+ no_merge */ empno, ename, sal,
      (select rpad(max(sal),10,' '))||
      rpad(min(sal),10,' ')||
      rpad(sum(sal),10,' ')||
      rpad(round(avg(sal)),10,' ')
      from emp) total
from emp);
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	1610	2 (0)	00:00:01
1	SORT AGGREGATE		1	13		
2	TABLE ACCESS FULL	EMP	14	182	2 (0)	00:00:01
3	VIEW		14	1610	2 (0)	00:00:01

	4		TABLE ACCESS FULL EMP		14		462		2	(0)	00:00:01	
--	---	--	------------------------	--	----	--	-----	--	---	-----	----------	--

0 db block gets
7 consistent gets
0 physical reads

문제 109. (점심시간 문제) 이름, 주소, 나이, 전공, 우리반 최대나이,
우리반 최소나이,
우리반 토탈나이,
우리반 평균나이를 출력하시오 !

풀이1 :

```
select ename, address, age, major, substr(total,1,10 ) 최대,
      substr(total,11,10 ) 최소,
      substr(total,21,10 ) 토탈,
      substr(total,31,10 ) 평균
from (select /*+ no_merge */ ename, address, age, major,
      (select rpad(max(age),10,' '))||
      rpad(min(age),10,' ')||
      rpad(sum(age),10,' ')||
      rpad(round(avg(age)),10,' ')
      from emp2) total
from emp2);
```

	Id		Operation		Name		Rows		Bytes		Cost (%CPU)		Time	
	0		SELECT STATEMENT				28		5768		2 (0)		00:00:01	
	1		SORT AGGREGATE				1		2					
	2		TABLE ACCESS FULL		EMP2		28		56		2 (0)		00:00:01	
	3		VIEW				28		5768		2 (0)		00:00:01	
	4		TABLE ACCESS FULL		EMP2		28		1988		2 (0)		00:00:01	

0 db block gets
6 consistent gets
0 physical reads

풀이2 :

```
select e.ename, e.address, e.age, e.major, t.최대, t.최소, t.토탈, t.평균
from emp2 e, (select max(age) 최대,
      min(age) 최소,
      sum(age) 토탈,
      round(avg(age)) 평균
      from emp2) t;
```

	Id		Operation		Name		Rows		Bytes		Cost (%CPU)		Time	
	0		SELECT STATEMENT				28		3444		4 (0)		00:00:01	
	1		NESTED LOOPS				28		3444		4 (0)		00:00:01	
	2		VIEW				1		52		2 (0)		00:00:01	
	3		SORT AGGREGATE				1		2					
	4		TABLE ACCESS FULL		EMP2		28		56		2 (0)		00:00:01	
	5		TABLE ACCESS FULL		EMP2		28		1988		2 (0)		00:00:01	

0 db block gets
6 consistent gets
0 physical reads

문제 110. 아래의 SQL을 분석함수를 사용하지 않는 SQL로 변경하시오 !

튜닝 후 :

```
select empno, ename, sal, sum(sal) over (order by sal, empno asc) 누적치
from emp;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	462	3 (34)	00:00:01
1	WINDOW SORT		14	462	3 (34)	00:00:01
2	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

0 recursive calls
0 db block gets
3 consistent gets

튜닝 전 :

```
with emp_sal as (select rownum as rn, e.*
                  from (select empno, ename, sal
                        from emp
                        order by sal) e)
select e1.empno, e1.ename, e1.sal,
       (select sum(sal)
        from emp_sal e2
        where e2.rn <= e1.rn) 누적치
from emp_sal e1;
```

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT AGGREGATE	
* 2	VIEW	
3	TABLE ACCESS FULL	SYS_TEMP_0FD9D6606_356DC0
4	TEMP TABLE TRANSFORMATION	
5	LOAD AS SELECT	SYS_TEMP_0FD9D6606_356DC0
6	COUNT	
7	VIEW	
8	SORT ORDER BY	
9	TABLE ACCESS FULL	EMP
10	VIEW	
11	TABLE ACCESS FULL	SYS_TEMP_0FD9D6606_356DC0

8 db block gets
36 consistent gets
1 physical reads

문제 111. 아래의 분석함수를 이용한 SQL을 분석함수를 이용하지 않은 SQL로 변경하시오 !

튜닝후 :

```
select deptno, empno, ename, sal,
       sum(sal) over ( partition by deptno order by empno) 누적치
from emp;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	644	3 (34)	00:00:01
1	WINDOW SORT		14	644	3 (34)	00:00:01
2	TABLE ACCESS FULL	EMP	14	644	2 (0)	00:00:01

```

0 db block gets
3 consistent gets
0 physical reads

```

튜닝 전 :

```

select deptno, empno, ename, sal, (select sum(sal)
                                   from emp ee
                                   where ee.deptno = e.deptno
                                   and ee.empno <= e.empno) 누적치
from emp e
order by deptno, empno;

```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	644	3 (34)	00:00:01
1	SORT AGGREGATE		1	39		
* 2	TABLE ACCESS FULL	EMP	1	39	2 (0)	00:00:01
3	SORT ORDER BY		14	644	3 (34)	00:00:01
4	TABLE ACCESS FULL	EMP	14	644	2 (0)	00:00:01

```

0 db block gets
45 consistent gets
0 physical reads

```

문제 112. 아래의 SQL을 분석함수를 이용하지 않은 SQL로 변경하시오 !

튜닝 후 :

```

select empno, ename, sal, lag(sal, 1) over (order by sal) 이전행
from emp;

```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	462	3 (34)	00:00:01
1	WINDOW SORT		14	462	3 (34)	00:00:01
2	TABLE ACCESS FULL	EMP	14	462	2 (0)	00:00:01

```

0 db block gets
3 consistent gets
0 physical reads

```

튜닝 전 :

```

select empno, ename, sal,
       (select sal
        from (select rownum rn, e3.*
              from (select * from emp e order by sal) e3) e4
        where e4.rn+1 = e2.rn) 이전행
from (select rownum rn, e1.*
      from (select * from emp e order by sal) e1) e2;

```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	644	3 (34)	00:00:01
* 1	VIEW		14	364	3 (34)	00:00:01
2	COUNT					
3	VIEW		14	182	3 (34)	00:00:01
4	SORT ORDER BY		14	1218	3 (34)	00:00:01
5	TABLE ACCESS FULL	EMP	14	1218	2 (0)	00:00:01

	6		VIEW				14		644		3	(34)		00:00:01	
	7		COUNT												
	8		VIEW				14		462		3	(34)		00:00:01	
	9		SORT ORDER BY				14		1218		3	(34)		00:00:01	
	10		TABLE ACCESS FULL		EMP		14		1218		2	(0)		00:00:01	

0 db block gets
 45 consistent gets
 0 physical reads