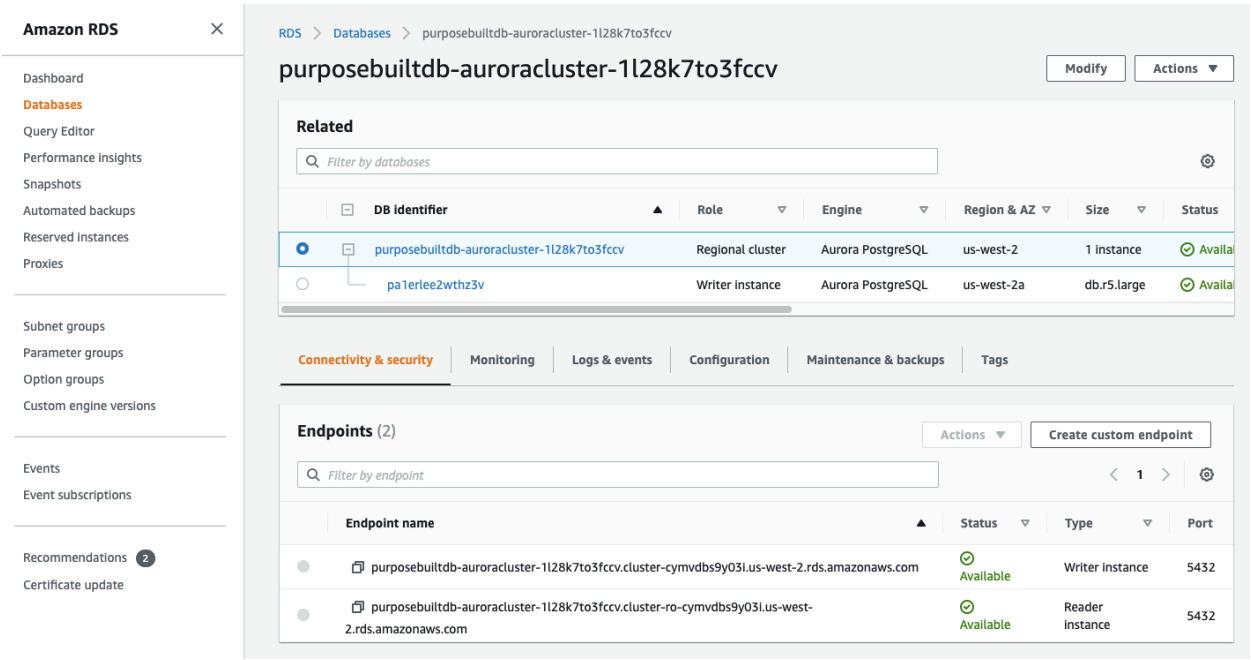
02) PostgreSQL 에서 초기데이터적재, 입력,수정,삭제, 조회 수행

[1] 선결 조건

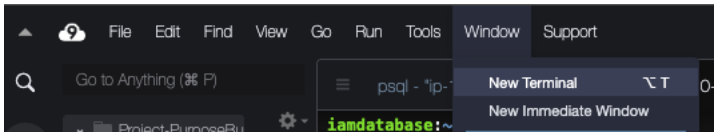
- 앞의 단계 완료

[2] psql 이용 데이터 적재

1. AWS Management console 돋보기에서 **RDS** 입력 → 새화면 Amazon RDS 에서 좌측 Databases 클릭 → 우측 상단에 생성된 DB identifier중 상단 purposebuiltdb-auroracluster... 을 클릭 → 하단의 2개 endpoint중 **writer instance endpoint 복사 및 저장** (이후 데이터베이스 접속시 사용)



2. AWS Management console → 돋보기에서 **cloud9** 입력 및 이동 → 새화면 AWS cloud9 에서 Project-PurposeBuiltDB 선택 → open IDE클릭 → 새화면 cloud9 에서 상단 window → new Terminal 을 클릭하여 새로운 terminal 을 오픈함



3. 생성된 terminal 을 이용해 데이터적재 수행

```
#shell에서 수행
cd ~/environment
export PGPASSWORD=auradmin123
psql -h <앞에서복사한 postgreSQL endpoint>  -U auradmin -d taxidb -p 5432


#postgreSQL의  taxidb=>  에서 수행
db  접속후 다음 명령 수행
\i ./builder202204/script/postgres-hr01.sql
\i ./builder202204/script/postgres-hr02.sql
```

4. 적재된 데이터를 조회

```
#postgreSQL의  taxidb=>  에서 수행
\d+ hr.departments
select * from hr.departments limit 5;

\d+ hr.employees
select * from hr.employees limit 5;
```

---

[3] 조회

1. 간단한 건 별 조회

```
#postgreSQL의  taxidb=>  에서 수행
# 대상 테이블 스캔
select * from hr.employees ;
select * from hr.departments ;

#사번 100번 인 직원정보인 직원정보를 조회
select a.*, b.*
from hr.employees a , hr.departments b
where a.department_id = b.department_id and a.employee_id = 100;

#이름의 first_name ='Steven' 이고 last_name = 'King' 인 직원정보를 조회
select a.*, b.*
from hr.employees a , hr.departments b
where a.department_id = b.department_id
 and a.first_name ='Steven' and  a.last_name = 'King';
```

2. Group by 조회

```
#postgreSQL의  taxidb=>  에서 수행
# 각 부서별  salary 합계 와 직원수
select department_id,sum(salary) as salary_sum, count(*) as cnt
from hr.employees
group by department_id
order by sum(salary) desc;
```

[4] 입력/수정/삭제

1. 입력

```
#postgreSQL의  taxidb=>  에서 수행
# 사번 300 번 신규사원 없음 확인
select * from hr.employees where employee_id = 300;

# 사번 300 번 신규사원 입력
insert into hr.employees values ( 300, 'joon','park','parkjoon', '010.1234.5678',CURRENT_DATE , 'AC_ACCOUNT', 1000.00,  null,   205 , 110 )

# 사번 300 번 신규사원 insert 확인
select * from hr.employees where employee_id = 300;
```

2. 수정

```
#postgreSQL의  taxidb=>  에서 수행
# 사번 300 번 사번의 변경전  직원 정보 조회
select * from hr.employees where employee_id = 300;

# 사번 300 번 사번의 salary 를 1000 -> 9999.9 로 update
update  hr.employees  set salary = 9999.9 where employee_id = 300;

# 사번 300 번 사번의 변경후 직원 정보 조회
select * from hr.employees where employee_id = 300;
```

3. 삭제

```
#postgreSQL의  taxidb=>  에서 수행
# 사번 300 번 사번의 delete 전  직원 정보 조회
select * from hr.employees where employee_id = 300;

# 사번 300 번 사번의 delete
delete from hr.employees  where employee_id = 300;

# 사번 300 번 사번의 delete 후  직원 정보 조회
select * from hr.employees where employee_id = 300;
```

[5] 부가 기능

1. 계층형 쿼리

```
#postgreSQL의  taxidb=>  에서 수행

WITH RECURSIVE a AS ( select employee_id, manager_id, first_name , job_id, 1::integer recursion_level
from hr.employees where employee_id = 100
union all
```

```
select d.employee_id, d.manager_id, d.first_name , d.job_id, a.recursion_level +1 from hr.employees d
JOIN a ON a.employee_id = d.manager_id)
select employee_id, manager_id, lpad(' ', recursion_level) ||first_name , job_id, recursion_level from a;


결과 검증용
select employee_id, manager_id, first_name , job_id   from hr.employees where employee_id in (206, 205, 101, 100 ) order by 1;
select employee_id, manager_id, first_name , job_id   from hr.employees where employee_id in ( 113,108, 101, 100)  order by 1;
```

---

[6] 데이터베이스, 테이블, 컬럼 정보

1. 터미널에서 데이터베이스 정보를 확인

```
#postgreSQL의  taxidb=>  에서 수행
select * from pg_catalog.pg_database  ;
\l+
```

2. 터미널에서 테이블 정보를 확인

```
#postgreSQL의  taxidb=>  에서 수행
select * from pg_catalog.pg_tables where schemaname = 'hr';
select * from information_schema.tables where table_schema = 'hr';
\dt hr.
```

3. 터미널에서 컬럼 정보를 확인

```
#postgreSQL의  taxidb=>  에서 수행
select * from information_schema.columns where table_schema = 'hr';
\dS hr.
```

4. 터미널에서 인덱스 정보를 확인

```
#postgreSQL의  taxidb=>  에서 수행
select * from pg_catalog.pg_indexes  where schemaname ='hr';
\di hr.
```

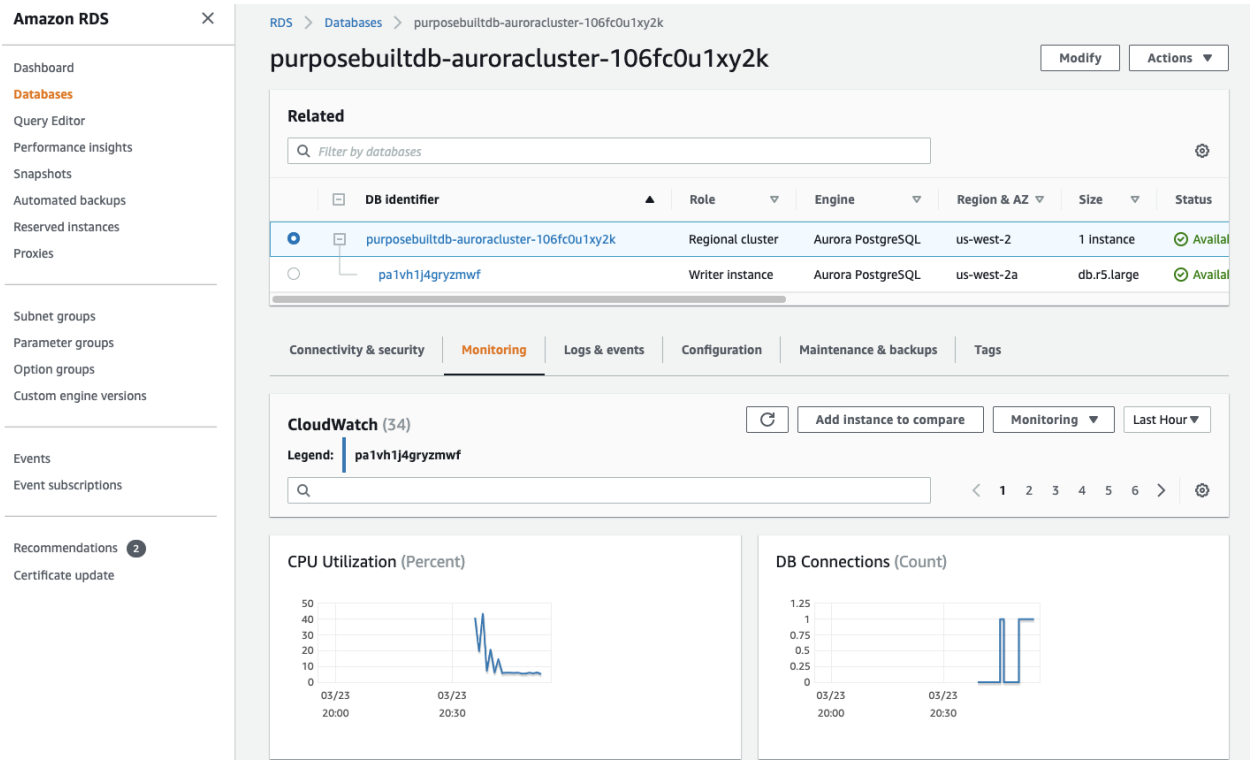
5. 터미널에서 관련 정보 모를때 시작점

```
#postgreSQL의  taxidb=>  에서 수행
select  table_catalog , table_schema||'.'||table_name as tname , table_type  from information_schema.tables  where table_name like '%유형(예,
\?
```

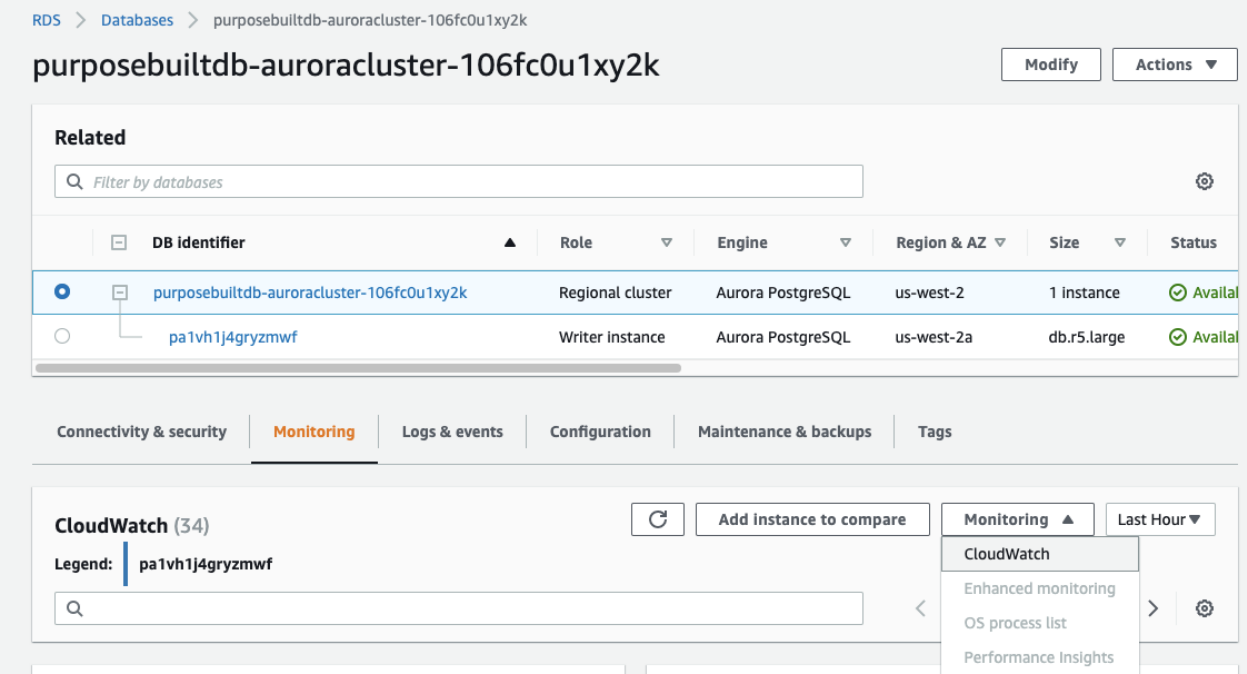
---

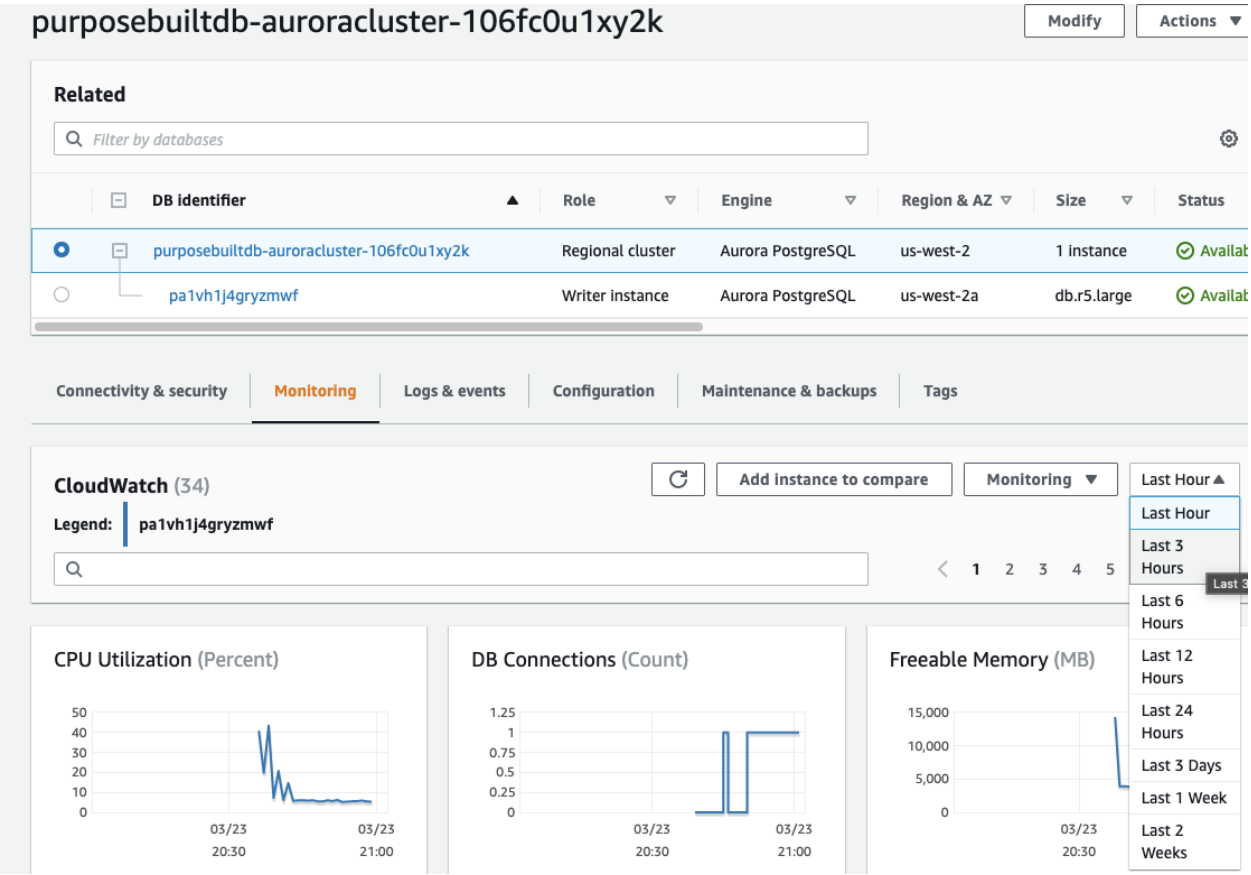
[7] 모니터링

1. RDS 화면 → 좌측 메뉴 에서 ‘Databases’ 선택 → 우측 화면에서 ‘purposebuiltdb-auroracluster...’ 선택 → 화면 중단의 ‘Monitoring ’ 탭을 선택하여 다양한 지표로 모니터링 (대상 인스턴스 단위 모니터링 가능)

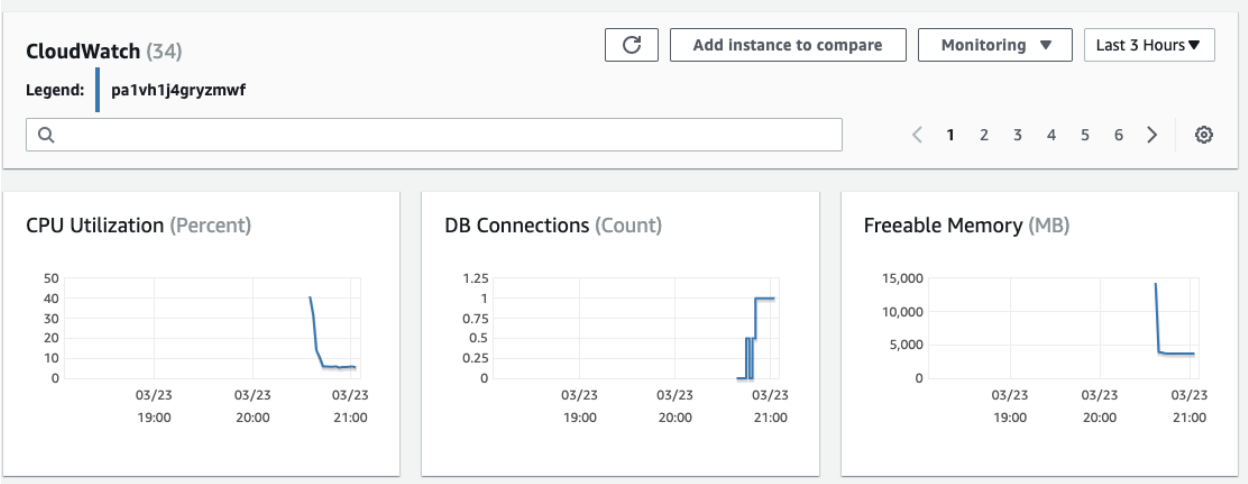


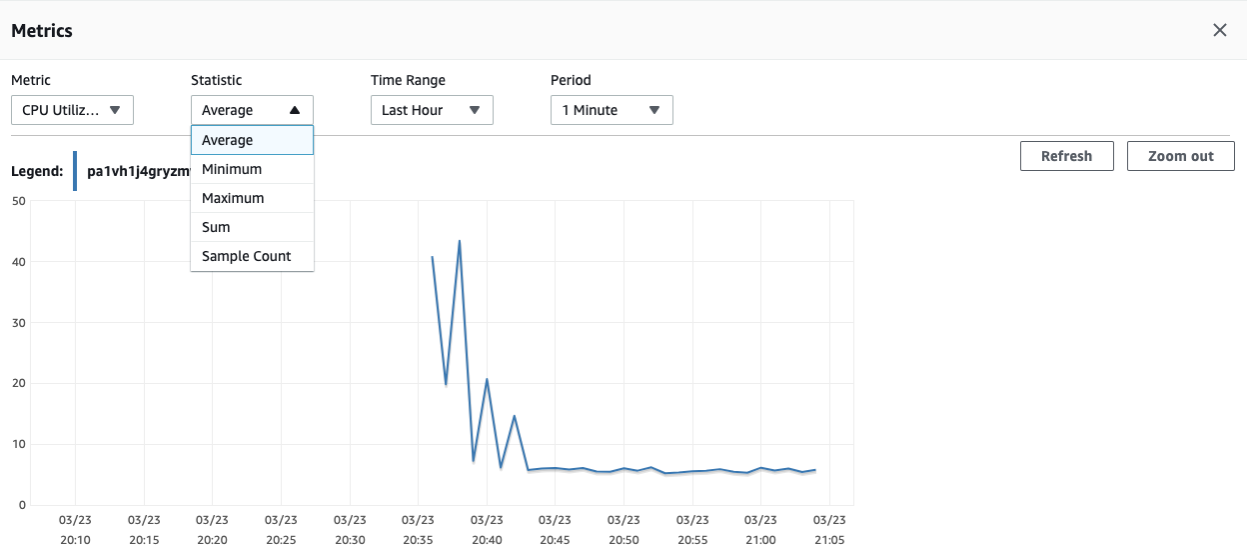
2. 화면 우측 중단의 monitoring 클릭 —> cloudwatch 선택 후 —> 우측 ‘Last Hour’에서 last 3 hours로 변경 (원하는 분석 시점 선택)


RDS > Databases > purposebuiltdb-auroracluster-106fc0u1xy2k
purposebuiltdb-auroracluster-106fc0u1xy2k
Modify
Actions
Related
Filter by databases
DB identifier
Role
Engine
Region & AZ
Size
Status
purposebuiltdb-auroracluster-106fc0u1xy2k
Regional cluster
Aurora PostgreSQL
us-west-2
1 instance
pa1vh1j4gryzmwf
Writer instance
Aurora PostgreSQL
us-west-2a
db.r5.large
Connectivity & security
Monitoring
Logs & events
Configuration
Maintenance & backups
Tags
CloudWatch (34)
Add instance to compare
Monitoring
Last Hour
Legend:
pa1vh1j4gryzmwf
CloudWatch
Enhanced monitoring
OS process list
Performance Insights


purposebuiltdb-auroracluster-106fc0u1xy2k
Modify
Actions
Related
Filter by databases
DB identifier
Role
Engine
Region & AZ
Size
Status
purposebuiltdb-auroracluster-106fc0u1xy2k
Regional cluster
Aurora PostgreSQL
us-west-2
1 instance
pa1vh1j4gryzmwf
Writer instance
Aurora PostgreSQL
us-west-2a
db.r5.large
Connectivity & security
Monitoring
Logs & events
Configuration
Maintenance & backups
Tags
CloudWatch (34)
Add instance to compare
Monitoring
Last Hour
Legend:
pa1vh1j4gryzmwf
1 2 3 4 5
Last Hour
Last 3 Hours
Last 6 Hours
Last 12 Hours
Last 24 Hours
Last 3 Days
Last 1 Week
Last 2 Weeks
CPU Utilization (Percent)
50
40
30
20
10
0
03/23 20:30
03/23 21:00
DB Connections (Count)
1.25
1
0.75
0.5
0.25
0
03/23 20:30
03/23 21:00
Freeable Memory (MB)
15,000
10,000
5,000
0
03/23 20:30

3. 최근 3시간 지표중에서 CPU Utilization 클릭 → 확대된 CPU Utilization 그래프 —> 보고자 하는 statistics, Time range, Period를 선택





- 수고하셨습니다. 다음 챕터로 이동하세요 —> 🗓️ 03) Dynamodb 에서 입력,수정,삭제,조회 수행