VLDB Lab.

Advanced SQL Practice

VLDB Lab.

Professor Sangwon Lee

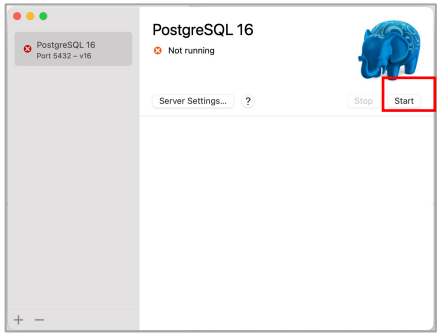
Contents

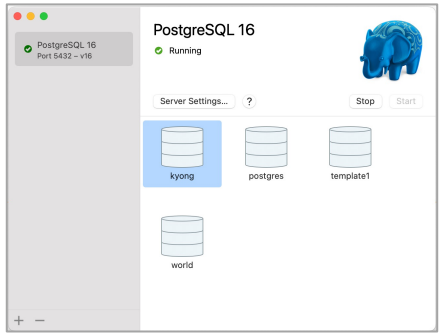
- **SCOTT schema 생성 (in postgres)**
- **Cube / Rollup**
- **Recursive**
- **Window Function and Analytic Function**

Start Postgres

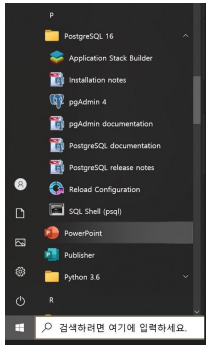
1. Start Postgres.

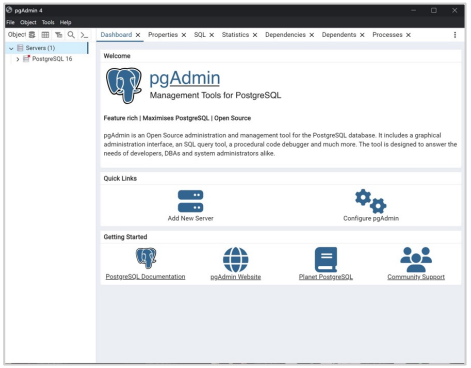
[Mac]



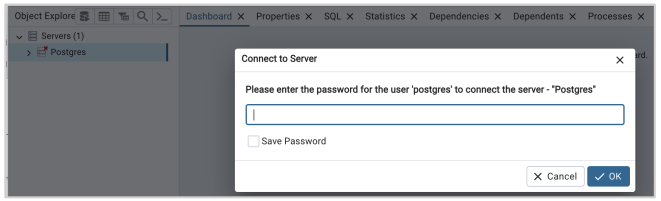


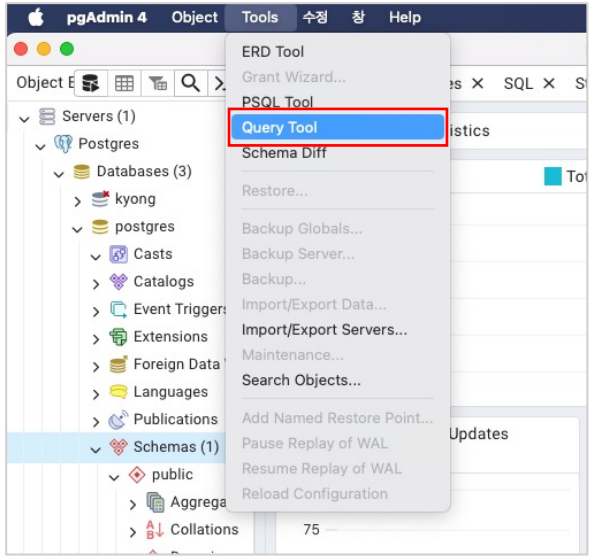
[Windows]

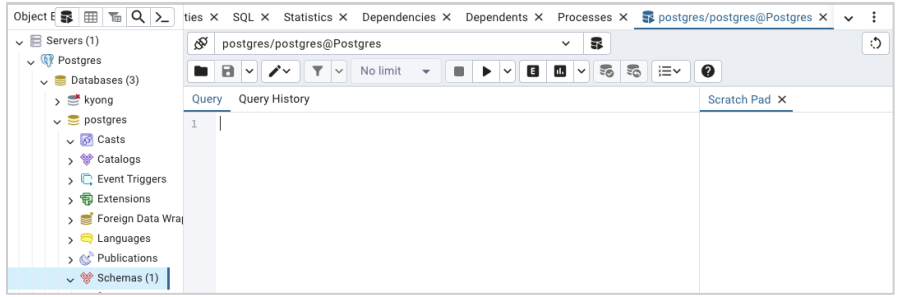




2. Start pgAdmin and connect postgres server.





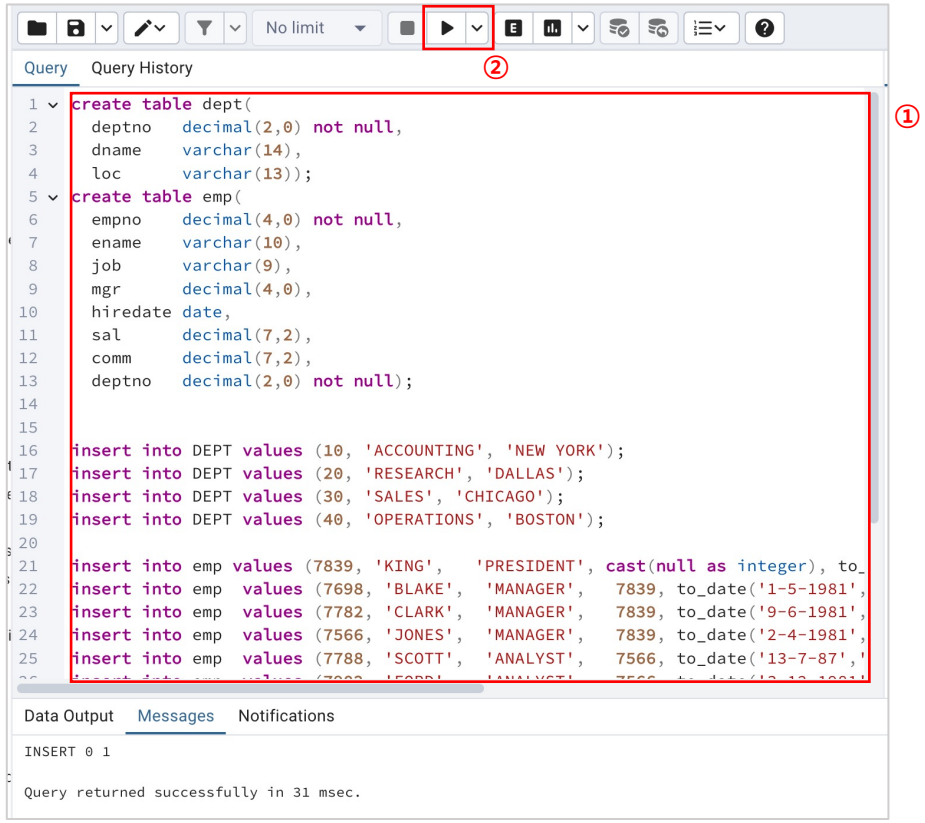


Now you can write sql query!

Postgres에서 SCOTT Schema 생성하기

1. SCOTT schema가 없는 경우, 다음 링크로 들어가서 scott schema를 복사합니다.
   https://github.com/kyongs/SNU-BigData-Fintech-F2024/blob/main/3/script.md

2. 복사한 sql문들을 postgres에서 실행합니다.



Cube/Rollup

ROLLUP

②

No limit

Query | Query History

①

```
SELECT job, deptno, sum(sal)
FROM emp
GROUP BY
ROLLUP(deptno,job);
```

Data Output | Messages | Notifications

	job character varying (9)	deptno numeric (2)	sum numeric
1	[null]	[null]	29025.00
2	ANALYST	20	3000.00
3	MANAGER	20	5425.00
4	MANAGER	30	2850.00
5	CLERK	20	1900.00
6	CLERK	30	950.00
7	SALESMAN	30	5600.00
8	ANALYST	10	3000.00
9	CLERK	10	1300.00
10	PRESIDENT	10	5000.00
11	[null]	10	9300.00
12	[null]	30	9400.00
13	[null]	20	10325.00

Cube

②

No limit

Query | Query History

①

```
SELECT deptno, job, sum(sal)
FROM emp
GROUP BY
CUBE(deptno,job);
```

Data Output | Messages | Notifications

	deptno numeric (2)	job character varying (9)	sum numeric
1	[null]	[null]	29025.00
2	20	ANALYST	3000.00
3	20	MANAGER	5425.00
4	30	MANAGER	2850.00
5	20	CLERK	1900.00
6	30	CLERK	950.00
7	30	SALESMAN	5600.00
8	10	ANALYST	3000.00
9	10	CLERK	1300.00
10	10	PRESIDENT	5000.00
11	10	[null]	9300.00
12	30	[null]	9400.00
13	20	[null]	10325.00
14	[null]	CLERK	4150.00
15	[null]	PRESIDENT	5000.00
16	[null]	MANAGER	8275.00
17	[null]	SALESMAN	5600.00
18	[null]	ANALYST	6000.00

다른 예제를 실행하기 위해 github 참고해 주세요!

Recursive

Recursive Example

②

No limit

Query Query History

①

```
with recursive VIEWNAME as(
    select 1 as num

    union all

    select num+1 from VIEWNAME where num < 10
)select * from VIEWNAME;
```

Data Output Messages Notifications

	num integer
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

②

No limit

Query Query History

①

```
WITH RECURSIVE EmployeeHierarchy AS (
    -- 기본 쿼리: 최고 관리자 (상사가 없는 직원)
    SELECT EMPNO, ENAME, JOB, MGR, 1 AS LEVEL
    FROM EMP
    WHERE MGR IS NULL

    UNION ALL

    -- 재귀 쿼리: 부하 직원
    SELECT e.EMPNO, e.ENAME, e.JOB, e.MGR, eh.LEVEL + 1
    FROM EMP e
    JOIN EmployeeHierarchy eh ON e.MGR = eh.EMPNO
)
SELECT LEVEL, EMPNO, ENAME, JOB, MGR
FROM EmployeeHierarchy
ORDER BY LEVEL, ENAME;
```

Data Output Messages Notifications

	level integer	empno numeric (4)	ename character varying (10)	job character varying (9)	mgr numeric (4)
1	1	7839	KING	PRESIDENT	[null]
2	2	7698	BLAKE	MANAGER	7839
3	2	7782	CLARK	MANAGER	7839
4	2	7566	JONES	MANAGER	7839
5	3	7499	ALLEN	SALESMAN	7698
6	3	7902	FORD	ANALYST	7566
7	3	7900	JAMES	CLERK	7698
8	3	7654	MARTIN	SALESMAN	7698
9	3	7934	MILLER	CLERK	7782
10	3	7788	SCOTT	ANALYST	7566
11	3	7844	TURNER	SALESMAN	7698
12	3	7521	WARD	SALESMAN	7698
13	4	7876	ADAMS	CLERK	7788
14	4	7369	SMITH	CLERK	7902

Analytic Function

RANK 예시) EMP 테이블에서 직원의 직업별 및 전체 급여 순위

Query | Query History

```
SELECT JOB, ENAME, SAL,
       RANK( ) OVER (ORDER BY SAL DESC) ALL_RANK,
       RANK( ) OVER (PARTITION BY JOB ORDER BY SAL DESC) JOB_RANK
  FROM EMP;
```

Data Output | Messages | Notifications

	job character varying (9)	ename character varying (10)	sal numeric (7,2)	all_rank bigint	job_rank bigint
1	PRESIDENT	KING	5000.00	1	1
2	ANALYST	FORD	3000.00	2	1
3	ANALYST	SCOTT	3000.00	2	1
4	MANAGER	JONES	2975.00	4	1
5	MANAGER	BLAKE	2850.00	5	2
6	MANAGER	CLARK	2450.00	6	3
7	SALESMAN	ALLEN	1600.00	7	1
8	SALESMAN	TURNER	1500.00	8	2
9	CLERK	MILLER	1300.00	9	1
10	SALESMAN	WARD	1250.00	10	3
11	SALESMAN	MARTIN	1250.00	10	3
12	CLERK	ADAMS	1100.00	12	2
13	CLERK	JAMES	950.00	13	3
14	CLERK	SMITH	800.00	14	4

Analytic Function

RANK와 DENSE RANK 차이) EMP 테이블에서 직원의 직업별 및 전체 급여 순위

No limit

Query | Query History

```
SELECT JOB, ENAME, SAL
     , RANK( ) OVER (ORDER BY SAL DESC) RANK
     , DENSE_RANK( ) OVER (ORDER BY SAL DESC) DENSE_RANK
  FROM EMP;
```

Data Output | Messages | Notifications

	job character varying (9)	ename character varying (10)	sal numeric (7,2)	rank bigint	dense_rank bigint
1	PRESIDENT	KING	5000.00	1	1
2	ANALYST	SCOTT	3000.00	2	2
3	ANALYST	FORD	3000.00	2	2
4	MANAGER	JONES	2975.00	4	3
5	MANAGER	BLAKE	2850.00	5	4
6	MANAGER	CLARK	2450.00	6	5
7	SALESMAN	ALLEN	1600.00	7	6
8	SALESMAN	TURNER	1500.00	8	7
9	CLERK	MILLER	1300.00	9	8
10	SALESMAN	WARD	1250.00	10	9
11	SALESMAN	MARTIN	1250.00	10	9
12	CLERK	ADAMS	1100.00	12	10
13	CLERK	JAMES	950.00	13	11
14	CLERK	SMITH	800.00	14	12

Analytic Function

Analytic Fuction) partition별 window 통계, 아래 예시는 SUM 함수

```
SELECT MGR, ENAME, SAL
     , SUM(SAL) OVER (PARTITION BY MGR) MGR_SUM
  FROM EMP;
```

	mgr numeric (4)	ename character varying (10)	sal numeric (7,2)	mgr_sum numeric
1	7566	FORD	3000.00	6000.00
2	7566	SCOTT	3000.00	6000.00
3	7698	TURNER	1500.00	6550.00
4	7698	ALLEN	1600.00	6550.00
5	7698	WARD	1250.00	6550.00
6	7698	MARTIN	1250.00	6550.00
7	7698	JAMES	950.00	6550.00
8	7782	MILLER	1300.00	1300.00
9	7788	ADAMS	1100.00	1100.00
10	7839	JONES	2975.00	8275.00
11	7839	CLARK	2450.00	8275.00
12	7839	BLAKE	2850.00	8275.00
13	7902	SMITH	800.00	800.00
14	[null]	KING	5000.00	5000.00

```
SELECT MGR, ENAME, SAL
     , SUM(SAL) OVER (PARTITION BY MGR ORDER BY SAL RANGE UNBOUNDED PRECEDING) as MGR_SUM
  FROM EMP;
```

	mgr numeric (4)	ename character varying (10)	sal numeric (7,2)	mgr_sum numeric
1	7566	SCOTT	3000.00	6000.00
2	7566	FORD	3000.00	6000.00
3	7698	JAMES	950.00	950.00
4	7698	MARTIN	1250.00	3450.00
5	7698	WARD	1250.00	3450.00
6	7698	TURNER	1500.00	4950.00
7	7698	ALLEN	1600.00	6550.00
8	7782	MILLER	1300.00	1300.00
9	7788	ADAMS	1100.00	1100.00
10	7839	CLARK	2450.00	2450.00
11	7839	BLAKE	2850.00	5300.00
12	7839	JONES	2975.00	8275.00
13	7902	SMITH	800.00	800.00
14	[null]	KING	5000.00	5000.00

Analytic Function

Analytic Fuction) COUNT 함수

Query | Query History

```
SELECT ENAME, SAL
     , COUNT(*) OVER
    (ORDER BY SAL RANGE BETWEEN 50 PRECEDING AND 150 FOLLOWING) as SIM_CNT
  FROM EMP;
```

Data Output | Messages | Notifications

	ename character varying (10)	sal numeric (7,2)	sim_cnt bigint
1	SMITH	800.00	2
2	JAMES	950.00	2
3	ADAMS	1100.00	3
4	WARD	1250.00	3
5	MARTIN	1250.00	3
6	MILLER	1300.00	3
7	TURNER	1500.00	2
8	ALLEN	1600.00	1
9	CLARK	2450.00	1
10	BLAKE	2850.00	4
11	JONES	2975.00	3
12	FORD	3000.00	3
13	SCOTT	3000.00	3
14	KING	5000.00	1

Analytic Function

First Value) 파티션별 윈도우에서 가장 먼저 나온 값

Query | Query History

```
SELECT DEPTNO, ENAME, SAL
     , FIRST_VALUE(ENAME) OVER (PARTITION BY DEPTNO ORDER BY SAL DESC ROWS UNBOUNDED PRECEDING) as DEPT_RICH
  FROM EMP;
```

Data Output | Messages | Notifications

	deptno numeric (2)	ename character varying (10)	sal numeric (7,2)	dept_rich character varying
1	10	KING	5000.00	KING
2	10	FORD	3000.00	KING
3	10	MILLER	1300.00	KING
4	20	SCOTT	3000.00	SCOTT
5	20	JONES	2975.00	SCOTT
6	20	CLARK	2450.00	SCOTT
7	20	ADAMS	1100.00	SCOTT
8	20	SMITH	800.00	SCOTT
9	30	BLAKE	2850.00	BLAKE
10	30	ALLEN	1600.00	BLAKE
11	30	TURNER	1500.00	BLAKE
12	30	MARTIN	1250.00	BLAKE
13	30	WARD	1250.00	BLAKE
14	30	JAMES	950.00	BLAKE

Analytic Function

PERCENT_RANK) 파티션별 윈도우에서 제일 먼저 나오는 것을 0으로 제일 늦게 나오는 것을 1로 하여 행 순서별 백분율 구함

No limit

Query | Query History

```
1 SELECT DEPTNO, ENAME, SAL
2       , PERCENT_RANK() OVER (PARTITION BY DEPTNO ORDER BY SAL DESC) as P_R
3   FROM EMP;
```

Data Output | Messages | Notifications

	deptno numeric (2)	ename character varying (10)	sal numeric (7,2)	p_r double precision
1	10	KING	5000.00	0
2	10	FORD	3000.00	0.5
3	10	MILLER	1300.00	1
4	20	SCOTT	3000.00	0
5	20	JONES	2975.00	0.25
6	20	CLARK	2450.00	0.5
7	20	ADAMS	1100.00	0.75
8	20	SMITH	800.00	1
9	30	BLAKE	2850.00	0
10	30	ALLEN	1600.00	0.2
11	30	TURNER	1500.00	0.4
12	30	MARTIN	1250.00	0.6
13	30	WARD	1250.00	0.6
14	30	JAMES	950.00	1

Analytic Function

CUME_DIST) 파티션별 위도우의 전체건수에서 현재 행보다 작거나 같은 건수에 대한 누적백분율을 구함

No limit

Query | Query History

```
1 SELECT DEPTNO, ENAME, SAL
2      , CUME_DIST() OVER (PARTITION BY DEPTNO ORDER BY SAL DESC) as CUME_DIST
3   FROM EMP;
```

Data Output | Messages | Notifications

	deptno numeric (2)	ename character varying (10)	sal numeric (7,2)	cume_dist double precision
1	10	KING	5000.00	0.3333333333333333
2	10	FORD	3000.00	0.6666666666666666
3	10	MILLER	1300.00	1
4	20	SCOTT	3000.00	0.2
5	20	JONES	2975.00	0.4
6	20	CLARK	2450.00	0.6
7	20	ADAMS	1100.00	0.8
8	20	SMITH	800.00	1
9	30	BLAKE	2850.00	0.16666666666666666
10	30	ALLEN	1600.00	0.3333333333333333
11	30	TURNER	1500.00	0.5
12	30	MARTIN	1250.00	0.8333333333333334
13	30	WARD	1250.00	0.8333333333333334
14	30	JAMES	950.00	1

Analytic Function

NTILE) 파티션별 전체 건수를 argument 값으로 n등분한 결과를 구함.

No limit

Query | Query History

```
SELECT ENAME, SAL
     , NTILE(4) OVER (ORDER BY SAL DESC) as QUAR_TILE
  FROM EMP ;
```

Data Output | Messages | Notifications

	ename character varying (10)	sal numeric (7,2)	quar_tile integer
1	KING	5000.00	1
2	SCOTT	3000.00	1
3	FORD	3000.00	1
4	JONES	2975.00	1
5	BLAKE	2850.00	2
6	CLARK	2450.00	2
7	ALLEN	1600.00	2
8	TURNER	1500.00	2
9	MILLER	1300.00	3
10	WARD	1250.00	3
11	MARTIN	1250.00	3
12	ADAMS	1100.00	4
13	JAMES	950.00	4
14	SMITH	800.00	4