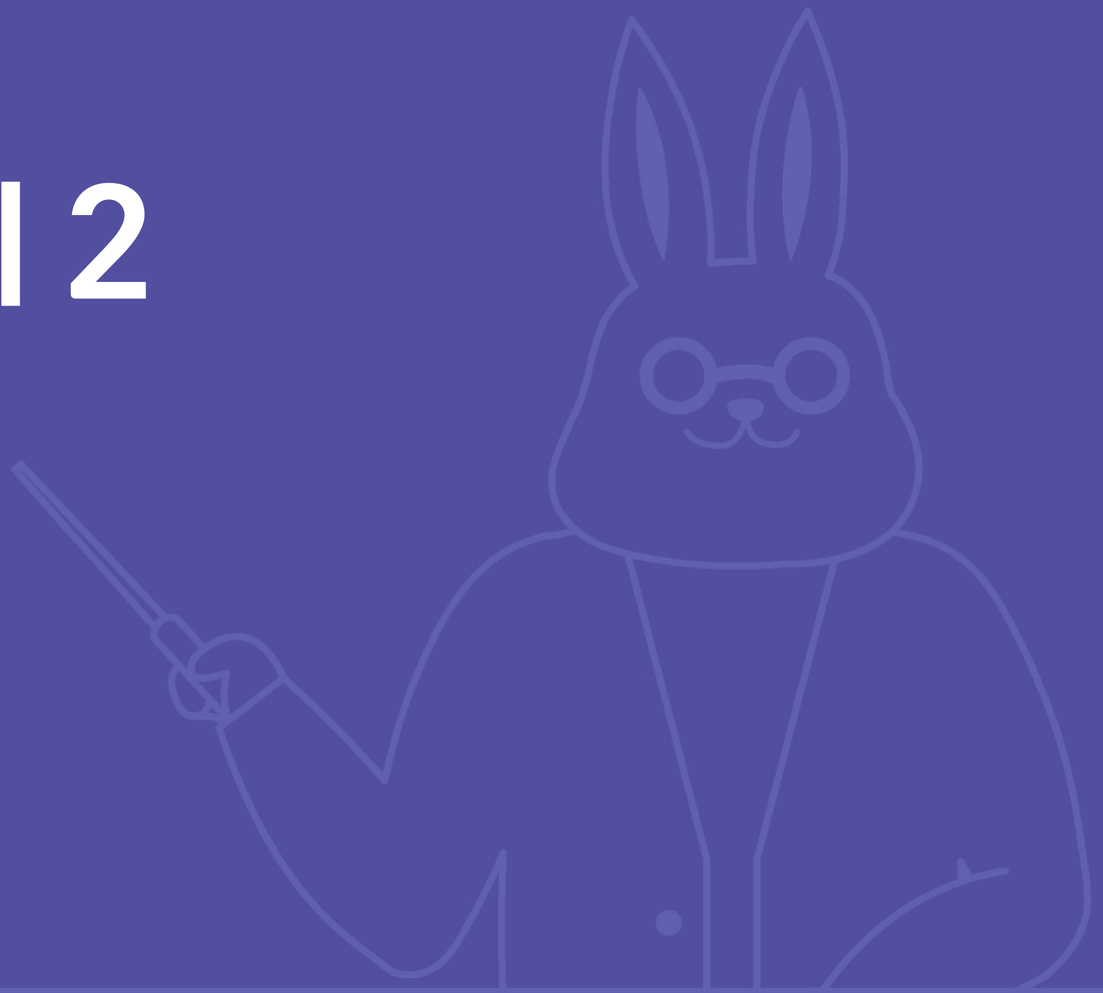
SQL로 데이터 다루기 2

04 그룹 함수 & 윈도우 함수

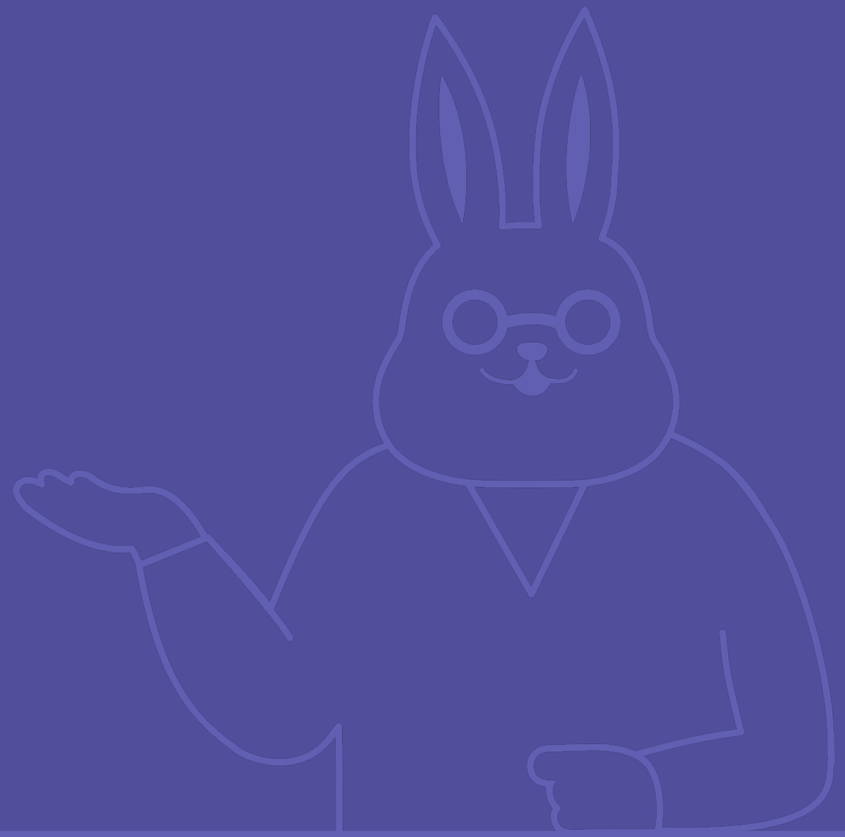


/* elice */

목차

01

데이터 분석 개요



데이터 분석을 위한 함수

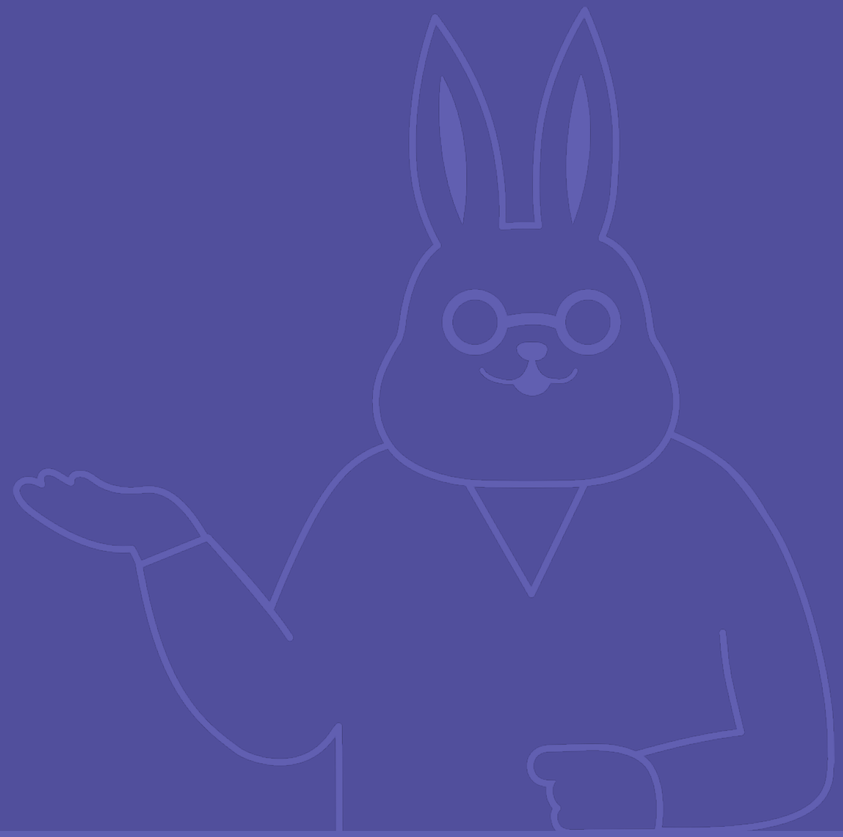
윈도우 함수
(Window fuction)

집계 함수
(Aggregate function)

그룹 함수
(Group function)

02

윈도우 함수



윈도우 함수

순위, 집계 등 행과 행 사이의 관계를 정의하는 함수

OVER 구문을 필수로 한다

SELECT WINDOW_FUNCTION (ARGUMENTS)

OVER ([PARTITION BY 칼럼] [ORDER BY 절] [WINDOWING 절]) FROM 테이블 명;

DBMS	Version
MySQL	8.0.22 이상
MariaDB	10.2.0 이상

윈도우 함수

SELECT WINDOW_FUNCTION (ARGUMENTS)

OVER ([PARTITION BY 칼럼] [ORDER BY 절] [WINDOWING 절]) FROM 테이블 명;

구조	설명
ARGUMENTS	윈도우 함수에 따라서 필요한 인수
PARTITION BY	전체 집합에 대해 소그룹으로 나누는 기준
ORDER BY	소그룹에 대한 정렬 기준
WINDOWING	행에 대한 범위 기준

구조	설명
ROWS	물리적 단위로 행의 집합을 지정
UNBOUNDED PRECEDING	윈도우의 시작 위치가 첫 번째 행
UNBOUNDED FOLLOWING	윈도우의 마지막 위치가 마지막 행
CURRENT ROW	윈도우의 시작 위치가 현재 행

DBMS	Version
MySQL	8.0.22 이상
MariaDB	10.2.0 이상

순위 함수

RANK() OVER ([PARTITION BY 컬럼][ORDER BY 컬럼][WINDOWING 절])

함수	설명
RANK	동일한 값에는 동일한 순위를 부여
DENSE_RANK	RANK와 같이 같은 값에는 같은 순위를 부여하나 한 건으로 취급
ROW_NUMBER	동일한 값이라도 고유한 순위를 부여

순위 관련 함수

코드

```
SELECT
    ID,
    NAME,
    SALARY,
    RANK() OVER (ORDER BY SALARY DESC) RANK,
    DENSE_RANK() OVER (ORDER BY SALARY DESC) DENSED_RANKING,
    ROW_NUMBER() OVER (ORDER BY SALARY DESC) ROW_NUMBER
FROM
    EMPLOYEE;
```

같은 값에 대한 처리 전략이 다르다

EMPLOYEE

ID	NAME	SALARY	RANK	DENSE_RANK	ROW_NUMBER
1004	ELICE	10000	1	1	1
1000	JAMES	8000	2	2	2
1001	JESSICA	6000	3	3	3
1005	JANNET	6000	3	3	4
1002	STEVE	4000	5	4	5
1003	ROBERT	1500	6	5	6

일반 집계 함수

일반 집계 함수(SUM, AVG, MAX, MIN, …)를
GROUP BY 구문 없이 사용할 수 있다.

일반 집계 함수

코드

```
SELECT
      ID, NAME, SALARY, DEPARTMENT_ID,
      (SELECT AVG(SALARY)
      FROM EMPLOYEE B
      WHERE B.DEPARTMENT_ID = A.DEPARTMENT_ID) DEPARTMENT_AVG
FROM
      EMPLOYEE A
ORDER BY
      A.DEPARTMENT_ID;
```

```
SELECT
    ID,
    NAME,
    SALARY,
    AVG(SALARY) OVER (PARTITION BY DEPARTMENT_ID) DEPARTMENT_AVG
FROM
    EMPLOYEE;
```

EMPLOYEE

ID	NAME	SALARY	DEPARTMENT_ID	DEPARTMENT_AVG
1000	JAMES	8000	1	5875
1002	STEVE	4000	1	5875
1004	ELICE	10000	1	5875
1003	ROBERT	1500	1	5875
1005	JANNET	6000	2	6000
1001	JESSICA	6000	2	6000

그룹 내 행 순서 함수

함수	설명
FIRST_VALUE	가장 먼저 나온 값을 구한다
LAST_VALUE	가장 나중에 나온 값을 구한다
LAG	이전 X 번째 행을 가져온다
LEAD	이후 X 번째 행을 가져온다

FIRST_VALUE, LAST_VALUE

각 부서에서 급여를 가장 많이 받은 사람의 급여와, 가장 적게 받는 사람의 급여를 함께 출력

코드

```
SELECT
    ID, DEPARTMENT_ID, NAME, SALARY,
    FIRST_VALUE(SALARY) OVER(PARTITION BY DEPARTMENT_ID ORDER BY SALARY
    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) DEPARTMENT_MIN_SALARY ,
    LAST_VALUE(SALARY) OVER(PARTITION BY DEPARTMENT_ID ORDER BY SALARY
    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) DEPARTMENT_MAX_SALARY
FROM EMPLOYEE
ORDER BY ID;
```

FIRST_VALUE, LAST_VALUE

코드

```
SELECT
    ID, DEPARTMENT_ID, NAME, SALARY,
    FIRST_VALUE(SALARY)
    OVER(PARTITION BY DEPARTMENT_ID ORDER BY SALARY
    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
    AS DEPARTMENT_MIN_SALARY ,
    LAST_VALUE(SALARY)
    OVER(PARTITION BY DEPARTMENT_ID ORDER BY SALARY
    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
    AS DEPARTMENT_MAX_SALARY
FROM
    EMPLOYEE
ORDER BY
    ID;
```

EMPLOYEE

ID	DEPARTMENT_ID	NAME	SALARY	DEPARTMENT_MIN_SALARY	DEPARTMENT_MAX_SALARY
1000	1	JAMES	8000	1500	10000
1001	2	JESSICA	6000	6000	6000
1002	1	STEVE	4000	1500	10000
1003	1	ROBERT	1500	1500	10000
1004	1	ALICE	10000	6000	6000
1005	2	JANNET	6000	1500	10000

LAG, LEAD

본인 사원 번호 앞과 뒤에 해당하는 직원의 이름을 함께 출력한다

코드

```
SELECT
    ID,
    NAME,
    SALARY,
    LAG(NAME, 1) OVER(ORDER BY ID) PREV_EMPLOYEE_NAME,
    LEAD(NAME, 1) OVER(ORDER BY ID) AFTER_EMPLOYEE_NAME
FROM
    EMPLOYEE;
```

ID	NAME	SALARY	PREV_EMPLOYEE_NAME	AFTER_EMPLOYEE_NAME
1000	JAMES	8000	(null)	JESSICA
1001	JESSICA	6000	JAMES	STEVE
1002	STEVE	4000	JESSICA	ROBERT
1003	ROBERT	1500	STEVE	ELICE
1004	ELICE	10000	ROBERT	JANNET
1005	JANNET	6000	ELICE	(null)

그룹 내 비율 함수

함수	설명
RATIO_TO_REPORT	파티션 내 전체 SUM에 대한 비율을 구한다
PERCENT_RANK	파티션 내 순위를 백분율로 구한다
CUME_DIST	파티션 내 현재 행보다 작거나 같은 건들의 수 누적 백분율로 구한다
NTILE	파티션 내 행들을 N등분한 결과를 구한다

RATIO_TO_REPORT

직원 전체 급여의 합 중 각 행이 차지하는 비율을 출력

코드

```
SELECT
    ID,
    NAME,
    SALARY,
    SUM(SALARY) OVER() TOTAL_SALARY,
    RATIO_TO_REPORT(SALARY) OVER() RATIO_TO_REPORT
FROM
    EMPLOYEE;
```

ID	NAME	SALARY	PREV_EMPLOYEE_NAME	AFTER_EMPLOYEE_NAME
1000	JAMES	8000	(null)	JESSICA
1001	JESSICA	6000	JAMES	STEVE
1002	STEVE	4000	JESSICA	ROBERT
1003	ROBERT	1500	STEVE	ELICE
1004	ELICE	10000	ROBERT	JANNET
1005	JANNET	6000	ELICE	(null)

RATIO_TO_REPORT-MariaDB

현재 MariaDB에서 RATIO_TO_REPORT는 제공되지 않기에 명시적으로 구현 가능한다

코드

```
SELECT
    ID,
    NAME,
    SALARY,
    SUM(SALARY) OVER() TOTAL_SALARY,
    (SALARY / SUM(SALARY) OVER()) RATIO_TO_REPORT
FROM
    EMPLOYEE;
```

ID	NAME	SALARY	TOTAL_SALARY	RATIO_TO_REPORT
1000	JAMES	8000	35500	0.2254
1001	JESSICA	6000	35500	0.1690
1002	STEVE	4000	35500	0.1127
1003	ROBERT	1500	35500	0.0423
1004	ELICE	10000	35500	0.2817
1005	JANNET	600	35500	0.1690

PERCENT_RANK, CUME_DIST

PERCENT_RANK는 순위를 백분율로 나타내며 제일 높은 순위 행은 0, 가장 낮은 순위 행은 1을 가진다
CUME_DIST는 현재 행보다 같거나 낮은 값들을 가지는 행들의 누적 백분율 값을 나타낸다.

코드

```
SELECT
    ID,
    NAME,
    SALARY,
    PERCENT_RANK() OVER(ORDER BY SALARY DESC)
   AS PERCENT_RANK,
    ROUND(CUME_DIST() OVER(ORDER BY SALARY DESC),  4)
    AS CUME_DIST
FROM
    EMPLOYEE;
```

0 이상 1 이하 값 반환

0	NAME	SALARY	PERCENT_RANK	CUME_DIST
1004	ELICE	10000	0	0.1667
1000	JAMES	8000	0.2	0.3333
1001	JESSICA	6000	0.4	0.6667
1005	JANNET	6000	0.4	0.6667
1002	STEVE	4000	0.8	0.8333
1003	ROBERT	1500	1	1

0 초과 1 이하 값 반환

NTILE

급여에 따라 직원들을 세 그룹으로 분류

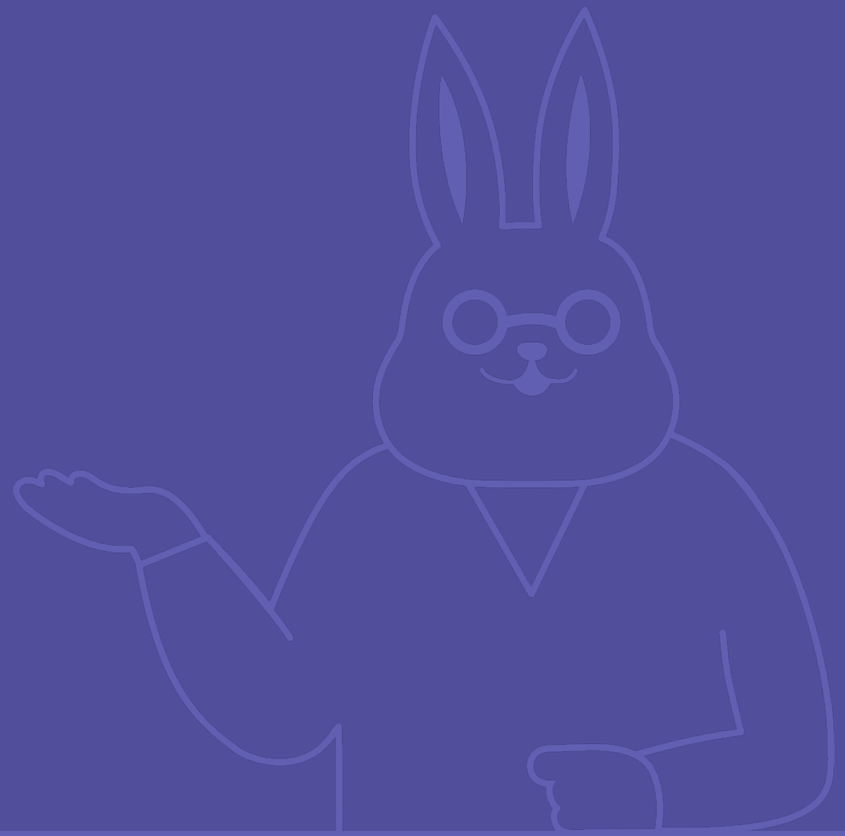
코드

```
SELECT
    ID,
    NAME,
    SALARY,
    NTILE(3) OVER(ORDER BY SALARY DESC) NTILE
FROM
    EMPLOYEE;
```

ID	NAME	SALARY	NTILE
1004	ELICE	10000	1
1000	JAMES	8000	1
1001	JESSICA	6000	2
1005	JANNET	6000	2
1002	STEVE	4000	3
1003	ROBERT	1500	3

03

그룹 함수



그룹 함수란?

데이터를 통계 내기 위해서는,

전체 데이터에 대한 통계는 물론이고 데이터 일부에 대한 소계, 중계 또한 필요하다.

각 레벨 별 SQL을 UNION문 으로 묶어 작성할 수도 있으나

ORACLE DB에서는 이러한 통계 데이터를 위한 몇 가지 함수를 제공한다

그룹 함수 – 기반 테이블

EMPLOYEE

ID	NAME	SALARY	DEPARTMENT_ID	JOB_ID
1000	JAMES	8000	1	5
1001	STEVE	4000	1	4
1002	LINDA	2500	1	3
1003	ROBERT	1500	1	1
1004	ELICE	10000	2	5
1005	TYRONE	1800	2	1
1006	BLAKE	3000	2	2
1007	DANIEL	5000	2	3
1008	SCOTT	5500	2	4
1009	MAXWELL	3000	1	2
1010	SCOTT	5500	1	4

JOB

JOB_ID	NAME
1000	JAMES
1001	STEVE
1002	LINDA
1003	ROBERT
1004	ELICE

DEPARTMENT

DEPARTMENT_ID	NAME
1000	JAMES
1001	STEVE

GROUP BY

코드

```
SELECT
    D.NAME AS DEPARTMENT_NAME,
        J.NAME AS JOB_NAME,
        AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E
JOIN DEPARTMENT D
ON E.DEPARTMENT_ID = D.ID
JOIN JOB J
ON E.JOB_ID = J.ID
GROUP BY
        D.NAME, J.NAME
ORDER BY D.NAME, J.NAME;
```

DEPARTMENT_NAME	JOB_NAME	AVG_SALARY
개발	과장	2500
개발	대리	3000
개발	부장	4750
개발	사원	1500
개발	팀장	8000
영업	과장	5000
영업	대리	3000
영업	부장	5500
영업	사원	1800
영업	팀장	10000

ROLL UP

그룹화하는 컬럼에 대한 부분적인 통계를 제공해준다

ROLL UP

코드

```
SELECT
    D.NAME AS DEPARTMENT_NAME,
        J.NAME AS JOB_NAME,
        AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E
JOIN DEPARTMENT D
ON E.DEPARTMENT_ID = D.ID
JOIN JOB J
ON E.JOB_ID = J.ID;
GROUP BY
        ROLLUP(D.NAME, J.NAME);
```

DEPARTMENT_NAME	JOB_NAME	AVG_SALARY
개발	과장	2500
개발	대리	3000
개발	부장	4750
개발	사원	1500
개발	팀장	8000
개발	(null)	4083.3333
영업	과장	5000
영업	대리	3000
영업	부장	5500
영업	사원	1800
영업	팀장	10000
영업	(null)	5060
(null)	(null)	4527.2727

ROLL UP – MariaDB

코드

```
SELECT
    D.NAME AS DEPARTMENT_NAME,
        J.NAME AS JOB_NAME,
        AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E
JOIN DEPARTMENT D
ON E.DEPARTMENT_ID = D.ID
JOIN JOB J
ON E.JOB_ID = J.ID;
GROUP BY
        D.NAME, J.NAME WITH ROLLUP;
```

DEPARTMENT_NAME	JOB_NAME	AVG_SALARY
개발	과장	2500
개발	대리	3000
개발	부장	4750
개발	사원	1500
개발	팀장	8000
개발	(null)	4083.3333
영업	과장	5000
영업	대리	3000
영업	부장	5500
영업	사원	1800
영업	팀장	10000
영업	(null)	5060
(null)	(null)	4527.2727

CUBE

ROLLUP 함수에서 제공하는 결과를 포함해서,

CUBE 함수에서는 그룹화 하는 컬럼에 대해

결합 가능한 모든 경우의 수에 대해 다차원 집계를 생성한다

CUBE

코드

```
SELECT
        D.NAME AS DEPARTMENT_NAME,
        J.NAME AS JOB_NAME,
        AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E
JOIN DEPARTMENT D ON E.DEPARTMENT_ID = D.ID
JOIN JOB J ON E.JOB_ID = J.ID
GROUP BY
        CUBE(D.NAME, J.NAME);
```

DEPARTMENT_NAME	JOB_NAME	AVG_SALARY
(null)	(null)	4527.2727
(null)	과장	3750
(null)	대리	3000
(null)	부장	5000
(null)	사원	1650
(null)	팀장	9000
개발	(null)	4083.3333
개발	과장	2500
개발	대리	3000
개발	부장	4750
개발	사원	1500
개발	팀장	8000
영업	(null)	5060
영업	과장	5000
영업	대리	3000
영업	부장	5500
영업	사원	1800
영업	팀장	10000

CUBE - MariaDB

코드

```
SELECT
        D.NAME AS DEPARTMENT_NAME,
        J.NAME AS JOB_NAME,
        AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E
JOIN DEPARTMENT D ON E.DEPARTMENT_ID = D.ID
JOIN JOB J ON E.JOB_ID = J.ID
GROUP BY
        D.NAME, J.NAME WITH ROLLUP
UNION
SELECT
        D.NAME AS DEPARTMENT_NAME,
        J.NAME AS JOB_NAME,
        AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E
JOIN DEPARTMENT D ON E.DEPARTMENT_ID = D.ID
JOIN JOB J ON E.JOB_ID = J.ID
GROUP BY
        J.NAME, D.NAME WITH ROLLUP;
```

DEPARTMENT_NAME	JOB_NAME	AVG_SALARY
개발	대리	3000.0000
개발	과장	2500.0000
개발	부장	4750.0000
개발	사원	1500.0000
개발	팀장	8000.0000
개발	(null)	4083.3333
영업	대리	3000.0000
영업	과장	5000.0000
영업	부장	5500.0000
영업	사원	1800.0000
영업	팀장	10000.0000
영업	(null)	5060.0000
(null)	(null)	4527.2727
(null)	대리	3000.0000
(null)	과장	3750.0000
(null)	부장	5000.0000
(null)	사원	1650.0000
(null)	팀장	9000.0000

GROUPING SETS

명시된 컬럼에 대해 개별 통계를 생성한다

각 컬럼에 대해 GROUP BY로 생성한 통계를 모두

UNION ALL한 결과와 동일하다

GROUPING SETS

코드

```
SELECT
        D.NAME AS DEPARTMENT_NAME,
        J.NAME AS JOB_NAME,
        AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E
JOIN DEPARTMENT D ON E.DEPARTMENT_ID = D.ID
JOIN JOB J ON E.JOB_ID = J.ID
GROUP BY
        GROUPING SETS(D.NAME, J.NAME);
```

DEPARTMENT_NAME	JOB_NAME	AVG_SALARY
영업	(null)	5060
개발	(null)	4083.3333
(null)	과장	3750
(null)	팀장	9000
(null)	대리	3000
(null)	부장	5000
(null)	사원	1650

GROUPING SETS

코드

```
SELECT
    D.NAME AS DEPARTMENT_NAME,
    NULL,
    AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E
JOIN DEPARTMENT D ON D.ID = E.DEPARTMENT_ID
GROUP BY D.NAME
UNION ALL
SELECT
    NULL, J.NAME AS JOB_NAME,
    AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E
JOIN JOB J ON J.ID = E.JOB_ID
GROUP BY J.NAME;
```

DEPARTMENT_NAME	JOB_NAME	AVG_SALARY
개발	(null)	4083.3333
영업	(null)	5060.0000
(null)	(null)	3000.0000
(null)	(null)	3750.0000
(null)	(null)	5000.0000
(null)	사원	1650.0000
(null)	팀장	9000.0000

크레딧

/* elice */

코스 매니저
하주희

콘텐츠 제작자
하주희, 문범우

강사
문범우

감수자
장석준

디자이너
강혜정

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io