실습 11. 분석함수(analytic function)

11-1. 각 사원의 급여와 소속부서 평균급여의 차이

```
SELECT
    성명,
    사원급여 - 소속부서평균급여 급여차이
FROM
    (
        SELECT
            a.first_name
            || ' '
            || a.last_name 성명,
            c.salary 사원급여,
            round(AVG(c.salary) OVER(
                PARTITION BY b.dept_no
            )) 소속부서평균급여
        FROM
            employees   a,
            dept_emp    b,
            salaries    c
        WHERE
            a.emp_no = b.emp_no
            AND b.TO_DATE = '99991231'
            AND a.emp_no = c.emp_no
            AND c.TO_DATE = '99991231'
    );
```

분석함수 avg() over ()

SQL formatted form by SQL Developer

출력 결과

	성명	급여차이
1	Shaowei Kitai	-5290
2	Fox Kugler	-16957
3	Zengping Peir	28542
4	Manton Selvestrel	17111
5	Mark Picht	-15161
6	Ashish Angelopoulos	-13649
7	Saniya Pepe	-574
8	Vincent Nergos	-21248
9	Bouchung Merlo	-19422
10	Sudharsan Langford	-2545
11	Masasuke Koprowski	28218
12	Morris DiGiano	-18672
13	Kasidit Cools	10932
14	Florina Koshiba	6390
15	Pasqua Kilgour	-11008
16	Fun Mawatari	26618
17	Mari Budinsky	-10906
18	Lucian Baak	-417
19	Lijia Litzkow	4620
20	Heon Thiran	14256
21	Goh Kaelbling	-1916
22	Udi Famili	-30872
23	Rildo Pepe	-2577

...

실습 11. 분석함수(analytic function)

11-2. 각 사원의 사번, 사원명, 입사일자, 급여변동일, 변동일당시급여, 급여인상액, 급여누적평균

```
select a.emp_no 사번
     , a.first_name||' '||a.last_name 성명
     , a.hire_date 입사일자
     , to_date(b.from_date, 'yyyymmdd') 급여변동일
     , b.salary 변동일당시급여
     , b.salary - lag(b.salary) over (partition by b.emp_no order by b.from_date) 급여인상액
     , round(avg(b.salary) over (partition by b.emp_no order by b.from_date)) 급여누적평균
from employees a, salaries b
where a.emp_no = b.emp_no ;
```

분석함수 lag() over ()

분석함수 avg() over ()

출력 결과

	사번	성명	입사일자	급여변동일	변동일당시급여	급여인상액	급여누적평균
1	10001	Georgi Facello	86/06/26	86/06/26	60117	(null)	60117
2	10001	Georgi Facello	86/06/26	87/06/26	62102	1985	61110
3	10001	Georgi Facello	86/06/26	88/06/25	66074	3972	62764
4	10001	Georgi Facello	86/06/26	89/06/25	66596	522	63722
5	10001	Georgi Facello	86/06/26	90/06/25	66961	365	64370
6	10001	Georgi Facello	86/06/26	91/06/25	71046	4085	65483
7	10001	Georgi Facello	86/06/26	92/06/24	74333	3287	66747
8	10001	Georgi Facello	86/06/26	93/06/24	75286	953	67814
9	10001	Georgi Facello	86/06/26	94/06/24	75994	708	68723
10	10001	Georgi Facello	86/06/26	95/06/24	76884	890	69539
11	10001	Georgi Facello	86/06/26	96/06/23	80013	3129	70491
12	10001	Georgi Facello	86/06/26	97/06/23	81025	1012	71369
13	10001	Georgi Facello	86/06/26	98/06/23	81097	72	72118
14	10001	Georgi Facello	86/06/26	99/06/23	84917	3820	73032
15	10001	Georgi Facello	86/06/26	00/06/22	85112	195	73837
16	10001	Georgi Facello	86/06/26	01/06/22	85097	-15	74541
17	10001	Georgi Facello	86/06/26	02/06/22	88958	3861	75389
18	10002	Bezalel Simmel	85/11/21	96/08/03	65828	(null)	65828
19	10002	Bezalel Simmel	85/11/21	97/08/03	65909	81	65869

...

개념 이해 11. 분석함수 (Analytic function)

분석함수 (analytic function)

- 테이블에 있는 데이터를 특정 용도로 분석하여 결과를 반환하는 함수
- 쿼리 결과Set을 대상으로 계산을 수행하는 함수
- SELECT 절에서 수행됨
- FROM, WHERE, GROUP BY 절에서 사용 불가
- ORDER BY 구문에서는 사용 가능

집계함수 vs. 분석함수

집계함수

집계함수는 여러행 또는 테이블 전체 행으로부터 그룹별로 집계하여 결과를 반환한다.

```
1  SELECT deptno
2       , SUM(sal) s_sal
3    FROM emp
4   GROUP BY deptno;
```

[그림] 집계함수 실행결과

DEPTNO	S_SAL
10	300
20	200
30	100

분석함수

분석 함수는 집계 결과를 각 행마다 보여준다.

```
1  SELECT deptno
2       , empno
3       , sal
4       , SUM(sal) OVER(PARTITION BY deptno) s_sal
5    FROM emp;
```

[그림] 분석함수 실행결과

DEPTNO	EMPNO	SAL	S_SAL
10	01	50	300
10	02	100	300
10	03	150	300
20	04	100	200
20	05	100	200
30	06	100	100

* 출처: http://www.gurubee.net/lecture/2671 [꿈꾸는 개발자, DBA커뮤니티 구루비]

개념 이해 11. 분석함수 (Analytic function)

집계함수 vs. 분석함수

- 집계함수는 그룹별 최대, 최소, 합계, 평균, 건수 등을 구할 때 사용되며, **그룹별 1개의 행을 반환**
- 분석함수는 그룹마다가 아니라 결과Set의 각 행마다 그룹별 계산결과를 보여줌

Syntax

```
1  SELECT ANALYTIC_FUNCTION ( arguments )
2         OVER ( [ PARTITION BY 컬럼List ]
3                [ ORDER BY 컬럼List ]
4                [ WINDOWING 절 (Rows|Range Between)]
5              )
6    FROM 테이블 명;
```

- ANALYTIC_FUNCTION : 분석함수명(입력인자)
- OVER : 분석함수임을 나타내는 키워드.
- PARTITION BY : 계산 대상 그룹을 정한다.
- ORDER BY : 대상 그룹에 대한 정렬을 수행한다.
- WINDOWING 절 : 분석함수의 계산 대상 범위를 지정한다.
  ORDER BY 절에 종속적이다.
  기본 생략 구문 : 정렬된 결과의 처음부터 현재행까지 [RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW]

분석함수의 종류

* 출처: http://www.gurubee.net/lecture/2671 [꿈꾸는 개발자, DBA커뮤니티 구루비]

- 순위함수 : RANK, DENSE_RANK, ROW_NUMBER, NTILE
- 집계함수 : SUM, MIN, MAX, AVG, COUNT
- 기타함수 : LEAD, LAG, FIRST_VALUE, LAST_VALUE, RATIO_TO_REPORT, KEEP, LISTAGG

실습 12. 행/열 전환(Pivoting)

12-1. 직급별 사원수 산출 : (행→열) 전환

12-1-① 직급별 사원수 산출을 위한 view 생성

```
create or replace view v_title_emp_rows
as
select title
     , count(*) cnt_emp
from titles
where to_date = '99991231'  -- 현재 기준
group by title ;
```

```
select * from v_title_emp_rows ;
```

출력 결과

	TITLE	CNT_EMP
1	Staff	25526
2	Manager	9
3	Engineer	30983
4	Technique Leader	12055
5	Assistant Engineer	3588
6	Senior Staff	82024
7	Senior Engineer	85939

12-1-② 직급에 대한 (행→열) 전환

```
select '사원수' 항목명
     , min(decode(title, 'Manager', cnt_emp)) Manager
     , min(decode(title, 'Technique Leader', cnt_emp)) Technique_Leader
     , min(decode(title, 'Senior Engineer', cnt_emp)) Senior_Engineer
     , min(decode(title, 'Engineer', cnt_emp)) Engineer
     , min(decode(title, 'Assistant Engineer', cnt_emp)) Assistant_Engineer
     , min(decode(title, 'Senior Staff', cnt_emp)) Senior_Staff
     , min(decode(title, 'Staff', cnt_emp)) Staff
from v_title_emp_rows ;
```

출력 결과

	항목명	MANAGER	TECHNIQUE_LEADER	SENIOR_ENGINEER	ENGINEER	ASSISTANT_ENGINEER	SENIOR_STAFF	STAFF
1	사원수	9	12055	85939	30983	3588	82024	25526

개념 이해 12. 뷰 (View)

- **뷰(view)**
  - 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 이름을 가지는 가상 테이블
  - 저장장치 내에 물리적으로 존재하지 않지만 사용자에게 있는 것처럼 간주되는 효과
  - 데이터 보정작업, 처리과정 시험 등 임시적인 작업을 위한 용도로 활용
  - 조인문의 사용 최소화로 사용상의 편의성을 최대화
- **특징**
  - 논리적 데이터 독립성을 제공한다.
  - 동일 데이터에 대해 동시에 여러사용자의 상이한 응용이나 요구를 지원해 준다.
  - 사용자의 데이터관리를 간단하게 해준다.
  - 접근 제어를 통한 자동 보안이 제공된다.

실습 12. 행/열 전환(Pivoting)

12-2. 직급별 사원수 산출 : (열→행) 전환

12-2-① 직급별 사원수 산출을 위한 view 생성 (12-1-①번 SQL을 view로 생성)

```
create or replace view v_title_emp_columns
as
select '사원수' 항목명
     , min(decode(title, 'Manager', cnt_emp)) Manager
     , min(decode(title, 'Technique Leader', cnt_emp)) Technique_Leader
     , min(decode(title, 'Senior Engineer', cnt_emp)) Senior_Engineer
     , min(decode(title, 'Engineer', cnt_emp)) Engineer
     , min(decode(title, 'Assistant Engineer', cnt_emp)) Assistant_Engineer
     , min(decode(title, 'Senior Staff', cnt_emp)) Senior_Staff
     , min(decode(title, 'Staff', cnt_emp)) Staff
from v_title_emp_rows ;

select * from v_title_emp_columns ;
```

출력 결과

	항목명	MANAGER	TECHNIQUE_LEADER	SENIOR_ENGINEER	ENGINEER	ASSISTANT_ENGINEER	SENIOR_STAFF	STAFF
1	사원수	9	12055	85939	30983	3588	82024	25526

실습 12. 행/열 전환(Pivoting)

12-2. 직급별 사원수 산출 : (열→행) 전환

12-2-② 직급에 대한 (열→행) 전환

```
select decode(no, 1, 'Manager'
                , 2, 'Technique Leader'
                , 3, 'Senior Engineer'
                , 4, 'Engineer'
                , 5, 'Assistant Engineer'
                , 6, 'Senior Staff'
                , 'Staff') title
     , decode(no, 1, Manager
                , 2, Technique_Leader
                , 3, Senior_Engineer
                , 4, Engineer
                , 5, Assistant_Engineer
                , 6, Senior_Staff
                , Staff) cnt_emp
from v_title_emp_columns a, copy_t b
where b.no <= 7 ;
```

출력 결과

	TITLE	CNT_EMP
1	Manager	9
2	Technique Leader	12055
3	Senior Engineer	85939
4	Engineer	30983
5	Assistant Engineer	3588
6	Senior Staff	82024
7	Staff	25526

실습 12. 행/열 전환(Pivoting)

12-2. 직급별 사원수 산출 : (열→행) 전환

12-2-③ With문 이용하기 (12-2-①번과 12-2-②번을 합쳐서 표현)

```
with title_emp_columns as
(select '사원수' 항목명
     , min(decode(title, 'Manager', cnt_emp)) Manager
     , min(decode(title, 'Technique Leader', cnt_emp)) Technique_Leader
     , min(decode(title, 'Senior Engineer', cnt_emp)) Senior_Engineer
     , min(decode(title, 'Engineer', cnt_emp)) Engineer
     , min(decode(title, 'Assistant Engineer', cnt_emp)) Assistant_Engineer
     , min(decode(title, 'Senior Staff', cnt_emp)) Senior_Staff
     , min(decode(title, 'Staff', cnt_emp)) Staff
from v_title_emp_rows)
select decode(no, 1, 'Manager'
                , 2, 'Technique Leader'
                , 3, 'Senior Engineer'
                , 4, 'Engineer'
                , 5, 'Assistant Engineer'
                , 6, 'Senior Staff'
                , 'Staff') title
     , decode(no, 1, Manager
                , 2, Technique_Leader
                , 3, Senior_Engineer
                , 4, Engineer
                , 5, Assistant_Engineer
                , 6, Senior_Staff
                , Staff) cnt_emp
from title_emp_columns a, copy_t b
where b.no <= 7 ;
```


실습 12. 행/열 전환(Pivoting)

12-3. 10-3번 예(부서별/직급별 급여합 산출)에 대한 (행->열) 전환

12-3-① 부서별/직급별 급여합에 대한 view 생성 (10-3번 SQL을 view로 생성)

```
create or replace view v_dept_title_salaries
as
select nvl(y.dept_name, '합계') 부서명
     , x.title 직급명
     , x.sum_sal 급여합
from ( select decode(grouping(b.dept_no), 1, '합계', b.dept_no) dept_no
            , decode(grouping(c.title), 1, '합계', c.title) title
            , sum(a.salary) sum_sal
       from salaries a, dept_emp b, titles c
       where a.emp_no = b.emp_no
         and b.emp_no = c.emp_no
         and a.to_date = '99991231' and b.to_date = '99991231' and c.to_date = '99991231'
       group by cube(b.dept_no, c.title) ) x, departments y
where x.dept_no = y.dept_no(+) ;
```

실습 12. 행/열 전환(Pivoting)

12-3. 10-3번 예(부서별/직급별 급여합 산출)에 대한 (행->열) 전환

12-3-① 부서별/직급별 급여합에 대한 view 생성 (10-3번 SQL을 view로 생성)

```
select * from v_dept_title_salaries ;
```

출력 결과

	부서명	직급명	급여합
1	Customer Service	Assistant Engineer	4013699
2	Customer Service	Technique Leader	16130639
3	Customer Service	Senior Engineer	127490191
4	Customer Service	Senior Staff	791929601
5	Customer Service	Engineer	37359981
6	Customer Service	Manager	58745
7	Customer Service	Staff	205151353
8	Customer Service	합계	1182134209
...			
55	합계	Assistant Engineer	205655454
56	합계	Technique Leader	813791946
57	합계	Senior Engineer	6086495408
58	합계	Senior Staff	6619869618
59	합계	Engineer	1846671624
60	합계	Manager	699513
61	합계	Staff	1718682560
62	합계	합계	17291866123

group by 부서, 직급

group by 부서

group by 직급

group by NULL

실습 12. 행/열 전환(Pivoting)

12-3. 10-3번 예(부서별/직급별 급여합 산출)에 대한 (행->열) 전환

12-3-② 직급에 대한 (행->열) 전환

```
select /*+ opt_param('_GBY_HASH_AGGREGATION_ENABLED' 'false') */ 부서명
     , min(decode(직급명, 'Assistant Engineer', 급여합)) "Assitant Engineer"
     , min(decode(직급명, 'Engineer', 급여합)) "Engineer"
     , min(decode(직급명, 'Senior Engineer', 급여합)) "Senior Engineer"
     , min(decode(직급명, 'Staff', 급여합)) "Staff"
     , min(decode(직급명, 'Senior Staff', 급여합)) "Senior Staff"
     , min(decode(직급명, 'Technique Leader', 급여합)) "Technique Leader"
     , min(decode(직급명, 'Manager', 급여합)) "Manager"
     , min(decode(직급명, '합계', 급여합)) "합계"
from v_dept_title_salaries
group by 부서명 ;
```

출력 결과

group by 부서, 직급

group by 부서

	부서명	Assitant Engineer	Engineer	Senior Engineer	Staff	Senior Staff	Technique Leader	Manager	합계
1	Customer Service	4013699	37359981	127490191	205151353	791929601	16130639	58745	1182134209
2	Development	95209078	838991757	2754762929	17601672	74353054	372256050	74510	4153249050
3	Finance	(null)	(null)	(null)	196491569	780474910	(null)	83457	977049936
4	Human Resources	(null)	(null)	(null)	165450013	658949251	(null)	65400	824464664
5	Marketing	(null)	(null)	(null)	247470796	940656147	(null)	106491	1188233434
6	Production	80311082	724966534	2394396077	19361437	77216162	320011423	56654	3616319369
7	Quality Management	21607713	195207616	650842874	(null)	(null)	84188157	72876	951919236
8	Research	4513882	50145736	159003337	166313637	647388761	21205677	79393	1048650423
9	Sales	(null)	(null)	(null)	700842083	2648901732	(null)	101987	3349845802
10	합계	205655454	1846671624	6086495408	1718682560	6619869618	813791946	699513	17291866123

group by 직급

group by NULL

실습 12. 행/열 전환(Pivoting)

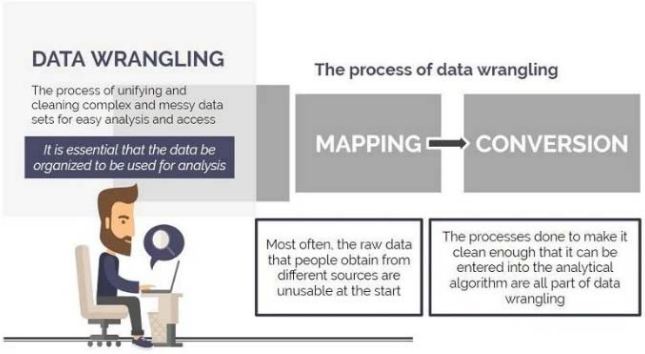
12-3. 10-3번 예(부서별/직급별 급여합 산출)에 대한 (행->열) 전환

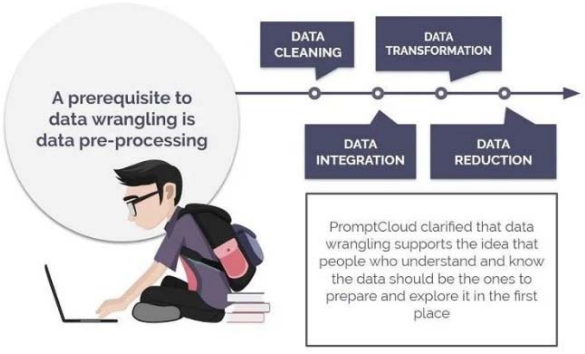
12-3-③ With문 이용하기 (12-3-①번과 12-3-②번을 합쳐서 표현)

```
with v_dept_title_salaries as
(select nvl(y.dept_name, '합계') 부서명
      , x.title 직급명
      , x.sum_sal 급여합
from ( select decode(grouping(b.dept_no), 1, '합계', b.dept_no) dept_no
            , decode(grouping(c.title), 1, '합계', c.title) title
            , sum(a.salary) sum_sal
       from salaries a, dept_emp b, titles c
       where a.emp_no = b.emp_no
         and b.emp_no = c.emp_no
         and a.to_date = '99991231' and b.to_date = '99991231' and c.to_date = '99991231'
       group by cube(b.dept_no, c.title) ) x, departments y
where x.dept_no = y.dept_no(+))
select /*+ opt_param('_GBY_HASH_AGGREGATION_ENABLED' 'false') */ 부서명
      , min(decode(직급명, 'Assistant Engineer', 급여합)) "Assitant Engineer"
      , min(decode(직급명, 'Engineer', 급여합)) "Engineer"
      , min(decode(직급명, 'Senior Engineer', 급여합)) "Senior Engineer"
      , min(decode(직급명, 'Staff', 급여합)) "Staff"
      , min(decode(직급명, 'Senior Staff', 급여합)) "Senior Staff"
      , min(decode(직급명, 'Technique Leader', 급여합)) "Technique Leader"
      , min(decode(직급명, 'Manager', 급여합)) "Manager"
      , min(decode(직급명, '합계', 급여합)) "합계"
from v_dept_title_salaries
group by 부서명 ;
```


개념 이해 13. Data Wrangling

- **데이터 랭글링(Data Wrangling) = 데이터 먼징(Data Munging)**
  - 복잡하고 지저분한 상태의 데이터를 간단한 분석과 접근을 위해 통합하는 과정
  - 데이터를 분석할 수 있는 상태로 조직하는 필수 과정 (데이터 전처리에서의 필수 과정)
  - 데이터 전처리 : 데이터 정제, 데이터 통합, 데이터 변형, 데이터 축소 등
  - Data Wrangling vs. ETL(Extraction/Transformation/Loading)
    - ✓ Data Wrangling : 분석가의 관점. 데이터를 준비하는 과정 (수동)
    - ✓ ETL : 최종 사용자 관점. 대부분 루틴한 과정 (자동)





* 출처: https://m.post.naver.com/viewer/postView.nhn?volumeNo=18028254&memberNo=43011790 [데이터 랭글링이 ... 이유]

개념 이해 14. With 문

With 문

- WITH 구문내의 쿼리의 결과(SUB쿼리)가 여러번 사용될때(호출될때) 유용
- 서브쿼리 블럭에 이름을 지정할 수 있도록 해 줌.
- 오라클 옵티마이저는 쿼리를 인라인뷰(inline view 방식)나 임시 테이블(materialized 방식)로 여김.
- Oracle 9 이상 지원

[WITH 구문 사용방법]

```
WITH ALIAS명 AS ( SUB쿼리 )
SELECT 컬럼명 FROM ALIAS명;
```

WITH 구문 예제)

```
WITH AA AS
  (SELECT ROWNUM, 'TEST1', SYSDATE
     FROM DUAL
   UNION ALL
   SELECT ROWNUM, 'TEST2', SYSDATE
     FROM DUAL
   UNION ALL
   SELECT ROWNUM, 'TEST3', SYSDATE FROM DUAL)
SELECT * FROM AA;
```

	ROWNUM	'TEST1'	SYSDATE
1	1	test1	2014-04-10 오후 2:39:55
2	1	test2	2014-04-10 오후 2:39:55
3	1	test3	2014-04-10 오후 2:39:55

[WITH 구문(2개 SUB쿼리) 사용방법]

```
WITH ALIAS명_1 AS ( SUB쿼리 ),
     ALIAS명_2 AS ( SUB쿼리 )
SELECT 컬럼명 FROM ALIAS명 where 조인조건;
```

WITH 구문(2개 SUB쿼리) 예제)

```
WITH AA AS
 (SELECT ROWNUM AS SEQ, 'TEST1' AS NAME, SYSDATE
    FROM DUAL
  UNION ALL
  SELECT ROWNUM AS SEQ, 'TEST2' AS NAME, SYSDATE
    FROM DUAL
  UNION ALL
  SELECT ROWNUM AS SEQ, 'TEST3' AS NAME, SYSDATE FROM DUAL),
BB AS
 (SELECT ROWNUM AS SEQ, 'TEST1' AS NAME, SYSDATE
    FROM DUAL
  UNION ALL
  SELECT ROWNUM AS SEQ, 'TEST2' AS NAME, SYSDATE
    FROM DUAL
  UNION ALL
  SELECT ROWNUM AS SEQ, 'TEST3' AS NAME, SYSDATE FROM DUAL)

SELECT * FROM AA, BB WHERE AA.NAME=BB.NAME
```

	SEQ	NAME	SYSDATE	SEQ	NAME	SYSDATE
1	1	test1	2014-04-10 오후 3:00:57	1	test1	2014-04-10 오후 3:00:57
2	1	test2	2014-04-10 오후 3:00:57	1	test2	2014-04-10 오후 3:00:57
3	1	test3	2014-04-10 오후 3:00:57	1	test3	2014-04-10 오후 3:00:57

* 출처: https://powerofwriting.tistory.com/entry/Oracle-WITH-구문-예제