

4-3. 집계 쿼리, 집합 연산자 활용 및 기타

홍형경

chariehong@gmail.com

2021.06

1. 집계 쿼리 실습

(1) 급여가 10000 이상인 사원의 평균 급여를 구하라

- 대상 테이블 : EMPLOYEES

- 급여가 10000 이상인 조건

 - ➔ salary >= 10000

- 평균 급여

 - ➔ AVG 집계 함수 사용

1. 집계 쿼리 실습

(1) 급여가 10000 이상인 사원의 평균 급여를 구하라

```
SELECT AVG(salary)  
FROM employees  
WHERE salary >= 10000;
```

AVG(SALARY)
12632.421052631578947368421...

```
SELECT ROUND(AVG(salary), 0)  
FROM employees  
WHERE salary >= 10000;
```

ROUND(AVG(SALARY),0)
12632

1. 집계 쿼리 실습

(2) 입사 월별 사원수를 구하라

- 대상 테이블 : EMPLOYEES

- 입사월

- 입사일자인 hire_date 컬럼 가공 필요
- TO_CHAR(hire_date, 'MM')

- 사원수

- COUNT 집계 함수 사용

1. 집계 쿼리 실습


(2) 입사 월별 사원수를 구하라

SELECT TO_CHAR(hire_date, 'MM')
 , COUNT(*)
FROM employees
GROUP BY TO_CHAR(hire_date, 'MM')
ORDER BY 1;

TO_CHAR(HIRE_DATE, 'MM')	COUNT(*)
01	14
02	13
03	17
04	7
05	6
06	11
07	7
08	9
09	5
10	6
11	5
12	7

1. 집계 쿼리 실습

(2-1) **요일별** 입사 인원수는?


SELECT TO_CHAR(hire_date, 'DAY')
 , COUNT(*)
FROM employees
GROUP BY TO_CHAR(hire_date, 'DAY')
ORDER BY 1;

TO_CHAR(HIRE_DATE, 'DAY')	COUNT(*)
월요일	19
화요일	16
수요일	15
목요일	10
금요일	15
토요일	19
일요일	13

1. 집계 쿼리 실습

(3) 이름이 동일한 사원과 동일인 수를 구하라

- 대상 테이블 : EMPLOYEES

- 이름

 - ➔ first_name 컬럼

- 사원수

 - ➔ COUNT 집계 함수 사용

1. 집계 쿼리 실습

(3) 이름이 동일한 사원과 동일인 수를 구하라

```
SELECT *  
FROM employees  
ORDER BY first_name;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
121	Adam	Fripp
196	Alana	Walsh
147	Alberto	Errazuriz
103	Alexander	Hunold
115	Alexander	Khoo
185	Alexis	Bull
158	Allan	McEwen
175	Alyssa	Hutton
167	Amit	Banda
187	Anthony	Cabrio
193	Britney	Everett
104	Bruce	Ernst
179	Charles	Johnson
153	Christopher	Olsen
162	Clara	Vishney
142	Curtis	Davies
109	Daniel	Faviet
163	Danielle	Greene
105	David	Austin
151	David	Bernstein
165	David	Lee
114	Den	Raphaely
107	Diana	Lorentz
198	Donald	OConnell

1. 집계 쿼리 실습

(3) 이름이 동일한 사원과 동일인 수를 구하라

SELECT first_name
 ,COUNT(*)
FROM employees
GROUP BY first_name
ORDER BY first_name;

FIRST_NAME	COUNT(*)
Adam	1
Alana	1
Alberto	1
Alexander	2
Alexis	1
Allan	1
Alyssa	1
Amit	1
Anthony	1
Britney	1
Bruce	1
Charles	1
Christopher	1
Clara	1
Curtis	1
Daniel	1
Danielle	1
David	3
Den	1
Diana	1
Donald	1
Douglas	1
Eleni	1
Elizabeth	1
Ellen	1
Gerald	1
Girard	1
Guy	1
Harrison	1
Hazel	1
Hermann	1

1. 집계 쿼리 실습

(3) 이름이 동일한 사원과 동일인 수를 구하라

```
SELECT first_name  
       ,COUNT(*)  
FROM employees  
GROUP BY first_name  
HAVING COUNT(*) > 1  
ORDER BY first_name;
```

FIRST_NAME	COUNT(*)
Alexander	2
David	3
James	2
Jennifer	2
John	3
Julia	2
Karen	2
Kevin	2
Michael	2
Peter	3
Randall	2
Steven	2
William	2

2. 집합연산자는 언제 사용할까?

- 어떤 상황에서 집합 연산자를 사용하는지에 대한 규칙은 없음
- 주어진 문제 해결을 위해 쿼리 작성을 하다 보면 자연스럽게 사용
- 명확한 규칙은 없으나 이러저러한 경우에 사용하는 경험 칙 소개

2. 집합연산자는 언제 사용할까?

(1) UNION (ALL)

- UNION 과 UNION ALL은 쓰임새가 같음 (중복 데이터 제거만 차이)
- 구조가 다른 여러 테이블에서 동일한 형태의 데이터를 추출하는 경우
- 컬럼을 로우 형태로 전환해 조회할 경우

2. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

· 예산 테이블 (budget_table)

```
create table budget_table (  
    yearmon    VARCHAR2(6),  
    budget_amt  NUMBER    );
```

```
INSERT INTO budget_table values('201901', 1000);  
INSERT INTO budget_table values('201902', 2000);  
INSERT INTO budget_table values('201903', 1500);  
INSERT INTO budget_table values('201904', 3000);  
INSERT INTO budget_table values('201905', 1050);
```

2. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

· 매출 테이블 (sale_table)

```
create table sale_table (  
    yearmon    VARCHAR2(6),  
    sale_amt   NUMBER    );
```

```
INSERT INTO sale_table values('201901', 900);  
INSERT INTO sale_table values('201902', 2000);  
INSERT INTO sale_table values('201903', 1000);  
INSERT INTO sale_table values('201904', 3100);  
INSERT INTO sale_table values('201905', 800);
```

2. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

```
SELECT *  
FROM budget_table;
```

```
SELECT *  
FROM sale_table;
```

	YEARMON	BUDGET_AMT
1	201901	1000
2	201902	2000
3	201903	1500
4	201904	3000
5	201905	1050

	YEARMON	SALE_AMT
1	201901	900
2	201902	2000
3	201903	1000
4	201904	3100
5	201905	800

2. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

* 계획 대비 실적은?

년월	계획
201901	1000
201902	2000
201903	1500

년월	실적
201901	900
201902	2000
201903	1000

년월	계획	실적	달성율
201901	1000	900	90%
201902	2000	2000	100%
201903	1500	1000	67%

2. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

```
SELECT yearmon, budget_amt, 0 sale_amt  
FROM budget_table
```

UNION

```
SELECT yearmon, 0 budget_amt, sale_amt  
FROM sale_table  
ORDER BY 1;
```

	YEARMON	BUDGET_AMT	SALE_AMT
1	201901	0	900
2	201901	1000	0
3	201902	0	2000
4	201902	2000	0
5	201903	0	1000
6	201903	1500	0
7	201904	0	3100
8	201904	3000	0
9	201905	0	800
10	201905	1050	0

2. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

```
SELECT yearmon,  
       SUM(budget_amt) budget,  
       SUM(sale_amt) sale,  
       ROUND(SUM(sale_amt) / SUM(budget_amt),2) * 100 rates  
FROM ( SELECT yearmon, budget_amt, 0 sale_amt  
      FROM budget_table  
      UNION  
      SELECT yearmon, 0 budget_amt, sale_amt  
      FROM sale_table  
      )  
GROUP BY yearmon  
ORDER BY 1;
```

	YEARMON	BUDGET	SALE	RATES
1	201901	1000	900	90
2	201902	2000	2000	100
3	201903	1500	1000	67
4	201904	3000	3100	103
5	201905	1050	800	76

2. 집합연산자는 언제 사용할까?

(1.2) UNION (ALL) – 컬럼을 로우로

```
CREATE TABLE test_score (
```

```
  years   VARCHAR2(4),    --년도  
  gubun   VARCHAR2(20),  --구분(중간,기말)  
  Korean  NUMBER,        --국어점수  
  english NUMBER,        --영어점수  
  math    NUMBER);       --수학점수
```

```
INSERT INTO test_score VALUES ('2019', '중간고사', 92, 87, 67);
```

```
INSERT INTO test_score VALUES ('2019', '기말고사', 88, 80, 91);
```

```
SELECT *
```

```
FROM test_score;
```

	YEARS	GUBUN	KOREAN	ENGLISH	MATH
1	2019	중간고사	92	87	67
2	2019	기말고사	88	80	91

2. 집합연산자는 언제 사용할까?

(1.2) UNION (ALL) – 컬럼을 로우로

SELECT years, gubun, '국어' subject, korean score

FROM test_score

UNION ALL

SELECT years, gubun, '영어' subject, english score

FROM test_score

UNION ALL

SELECT years, gubun, '수학' subject, math score

FROM test_score

ORDER BY 2 desc;

	YEARS	GUBUN	SUBJECT	SCORE
1	2019	중간고사	국어	92
2	2019	중간고사	영어	87
3	2019	중간고사	수학	67
4	2019	기말고사	영어	80
5	2019	기말고사	수학	91
6	2019	기말고사	국어	88

2. 집합연산자는 언제 사용할까?

(1.3) INTERSECT

→ **locations** 테이블에서 **city**와 **state_province** 값이 같은 값 조회

SELECT *

FROM locations;

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1600	2007 Zagora St	50090	South Brunswick	New Jersey	US
1700	2004 Charade Rd	98199	Seattle	Washington	US
1800	147 Spadina Ave	M5V 2L7	Toronto	Ontario	CA
1900	6092 Boxwood St	YSW 9T2	Whitehorse	Yukon	CA
2000	40-5-12 Laogianggen	190518	Beijing	(null)	CN
2100	1298 Vileparle (E)	490231	Bombay	Maharashtra	IN
2200	12-98 Victoria Street	2901	Sydney	New South Wales	AU
2300	198 Clementi North	540198	Singapore	(null)	SG
2400	8204 Arthur St	(null)	London	(null)	UK
2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK
2600	9702 Chester Road	09629850293	Stretford	Manchester	UK
2700	Schwanthalerstr. 7031	80925	Munich	Bavaria	DE

2. 집합연산자는 언제 사용할까?

(1.3) INTERSECT

```
SELECT state_province dup_loc_name
FROM locations
INTERSECT
SELECT city
FROM locations
ORDER BY 1;
```

	DUP_LOC_NAME
1	Oxford
2	Sao Paulo
3	Utrecht

2. 집합연산자는 언제 사용할까?

(1.3) INTERSECT

```
SELECT state_province, city  
FROM locations  
WHERE state_province = city  
ORDER BY 1;
```

	STATE_PROVINCE	CITY
1	Oxford	Oxford
2	Sao Paulo	Sao Paulo
3	Utrecht	Utrecht

3. 보너스 문제

- GROUPLYMULTIPLY 테이블 생성

```
create table GROUPLYMULTIPLY (  
  department_name VARCHAR2(100),  
  num_data        NUMBER  
);
```


3. 보너스 문제

· GROUPLYMULTIPLY 테이블 데이터 입력

```
insert into groupbymultiply values ('dept1', 10);  
insert into groupbymultiply values ('dept1', 20);  
insert into groupbymultiply values ('dept1', 30);  
insert into groupbymultiply values ('dept2', 5);  
insert into groupbymultiply values ('dept2', 7);  
insert into groupbymultiply values ('dept2', 40);  
insert into groupbymultiply values ('dept3', 69);  
insert into groupbymultiply values ('dept3', 71);  
insert into groupbymultiply values ('dept3', 12);
```

```
commit;
```

3. 보너스 문제

· GROUPBYMULTIPLY 테이블 조회

```
SELECT *  
FROM groupbymultiply;
```

	DEPARTMENT_NAME	NUM_DATA
1	dept1	10
2	dept1	20
3	dept1	30
4	dept2	5
5	dept2	7
6	dept2	40
7	dept3	69
8	dept3	71
9	dept3	12

3. 보너스 문제

· GROUPBYMULTIPLY 테이블 집계

```
SELECT department_name,  
       SUM(num_data)  
FROM groupbymultiply  
GROUP BY department_name  
ORDER BY 1 ;
```

	DEPARTMENT_NAME	NUM_DATA
1	dept1	10
2	dept1	20
3	dept1	30
4	dept2	5
5	dept2	7
6	dept2	40
7	dept3	69
8	dept3	71
9	dept3	12

	DEPARTMENT_NAME	SUM(NUM_DATA)
1	dept1	60
2	dept2	52
3	dept3	152

3. 보너스 문제

· GROUPBYMULTIPLY 테이블 문제

Department_name 별로 num_data 컬럼
값을 더하는 것이 아니라 곱한
결과를 조회하는 쿼리를 작성하시오.

	DEPARTMENT_NAME	NUM_DATA
1	dept1	10
2	dept1	20
3	dept1	30
4	dept2	5
5	dept2	7
6	dept2	40
7	dept3	69
8	dept3	71
9	dept3	12

	DEPARTMENT_NAME	MULTIPLY_RESULT
1	dept1	6000
2	dept2	1400
3	dept3	58788

3. 보너스 문제

- SUM 함수는 값을 모두 더함

- 곱하기를 더하기로 변환이 필요

- 로그의 덧셈 공식

- 로그 덧셈 → exp 함수를 사용해 변환

$$\log_e 2 = x \rightarrow e^x = 2$$

$$\log_e 3 = y \rightarrow e^y = 3$$

$$e^x * e^y = e^{x+y}$$

$$\log_e 2 + \log_e 3 = \log_e 2 * 3$$

$$\log(10 * 20 * 30) = \log 10 + \log 20 + \log 30$$

3. 보너스 문제

· GROUPBYMULTIPLY 테이블 집계

```
SELECT department_name
       ,EXP(SUM(LN(num_data))) multiply_result
FROM groupbymultiply
WHERE 1=1
GROUP BY department_name
ORDER BY 1;
```

	DEPARTMENT_NAME	NUM_DATA
1	dept1	10
2	dept1	20
3	dept1	30
4	dept2	5
5	dept2	7
6	dept2	40
7	dept3	69
8	dept3	71
9	dept3	12

	DEPARTMENT_NAME	MULTIPLY_RESULT
1	dept1	6000
2	dept2	1400
3	dept3	58788

[illegible]