

데이터베이스 4분반

HOMEWORK #3

1876217 컴퓨터공학과 안서연

목차

1. 숫자 데이터 요약 함수
2. CASE 조건문 함수
3. DECODE 함수
4. GROUP BY 함수
5. HAVING 함수

1.1 SELECT function (열 이름) FROM 테이블명; 형태

1.1 SELECT function (열 이름) FROM 테이블명; 형태

```
SELECT MAX(START_DATE),MIN(START_DATE)
FROM JOB_HISTORY;
```

JOB_HISTORY 참조 테이블에서 다른 조건 없이 START_DATE 열의 MAX(최댓값),MIN(최솟값)을 숫자 데이터 요약 함수를 통해서 값을 출력하였다.

```
SELECT AVG(MAX_SALARY), MAX(MAX_SALARY),
       MIN(MAX_SALARY), SUM(MAX_SALARY)
FROM JOBS
WHERE JOB_TITLE LIKE '%Manager%';
```

	AVG(MAX_SALARY)	MAX(MAX_SALARY)	MIN(MAX_SALARY)	SUM(MAX_SALARY)
1	15096.6666666666666666666666666667	20080	8500	90580

JOBS 참조테이블에서 JOB_TITLE에 Manager 이 포함된 행들을 대상으로 MAX_SALARY 의 MAX(최댓값), MIN(최솟값),SUM(합계)를 숫자 데이터 요약 함수를 통해서 값을 출력 하였다.

1. 숫자 데이터 요약 함수 (EX. COUNT)

1.2 SELECT COUNT (*) FROM 테이블명;

```
SELECT COUNT(*)  
FROM EMPLOYEES  
WHERE SALARY>8000;
```

<출력 결과>

COUNT(*)
1 33

COUNT(*)은 NULL 값을 포함한 테이블 행 개수를 출력하는 것으로, EMPLOYEES 참조테이블에서 SALARY 행의 값이 8000 초과하는 조건에 해당되는 행의 개수를 NULL 값을 포함하여 출력한 결과 33개임을 알 수 있다.

1.3 SELECT COUNT (열 이름) FROM 테이블명;

```
SELECT COUNT(MANAGER_ID)  
FROM DEPARTMENTS;
```

<출력 결과>

COUNT(MANAGER_ID)
1 11

COUNT(열 이름)은 NULL 값을 제외한 테이블 행 수를 출력하는 것으로, DEPARTMENTS 참조테이블에서 NULL 값이 아닌 MAGER_ID 값을 출력하였더니 11개가 출력되었다. DEPARTMENTS 는 총 27행이므로 16행의 MANAGER_ID가 NULL 값을 알 수 있다.

1.4 SELECT COUNT (DISTINCT 열 이름) FROM 테이블명;

```
SELECT COUNT(DISTINCT COUNTRY_ID)  
FROM LOCATIONS;
```

<출력 결과>

COUNT(DISTINCTCOUNTRY_ID)
1 14

COUNT(DISTINCT 열 이름)은 중복을 제거한 테이블 행을 출력하는 것으로, LOCATIONS 참조테이블에서 중복을 제거한 COUNTRY_ID 행의 개수를 출력하였더니 14개임을 알 수 있다. LOCATIONS 테이블은 총 23행으로 다른 행과 중복된 값이 9행 있었음을 알 수 있다.

2. CASE 조건문 함수

2.1 SELECT 열 이름

```
CASE WHEN [조건1] THEN [결과값1]
      WHEN [조건2] THEN [결과값2]
      WHEN [조건3] THEN [결과값3]
      ELSE [결과값4] END AS 새로운 열 이름
FROM 테이블명;
```

```
SELECT JOB_ID, COMMISSION_PCT, SALARY,
       CASE COMMISSION_PCT WHEN 0.4 THEN SALARY*1.2
                           WHEN 0.3 THEN SALARY*1.1
                           WHEN 0.25 THEN SALARY*1.05
                           ELSE SALARY END "REVISED_SALARY"
FROM EMPLOYEES
ORDER BY REVISED_SALARY;
```

EMPLOYEES 참조테이블에서
JOB_ID, COMMISSION_PCT, SALARY 를 선택하여
COMMISSION_PCT 가 0.4 이면 SALARY에 1.2 를
곱하고, 0.3 이면 SALARY에 1.1을 곱하고, 0.25이
면 SALARY 에 1.05를 곱하고 나머지는 기존
SALARY 값으로 설정하여 새로운
REVISED_SALARY 열에 변경된 값을 나타내어
JOB_ID, COMMISSION_PCT, SALARY 열과 함께 출
력하였다. 총 107행이 출력되었다.

<출력 결과>

JOB_ID	COMMISSION_PCT	SALARY	REVISED_SALARY
1 ST_CLERK	(null)	2100	2100
2 ST_CLERK	(null)	2200	2200
3 ST_CLERK	(null)	2200	2200
4 ST_CLERK	(null)	2400	2400
5 ST_CLERK	(null)	2400	2400
6 PU_CLERK	(null)	2500	2500
7 ST_CLERK	(null)	2500	2500
8 ST_CLERK	(null)	2500	2500
9 ST_CLERK	(null)	2500	2500
10 SH_CLERK	(null)	2500	2500
11 SH_CLERK	(null)	2500	2500
12 PU_CLERK	(null)	2600	2600
13 ST_CLERK	(null)	2600	2600
14 SH_CLERK	(null)	2600	2600
15 SH_CLERK	(null)	2600	2600
16 ST_CLERK	(null)	2700	2700
17 ST_CLERK	(null)	2700	2700
18 PU_CLERK	(null)	2800	2800

...

90 PR_REP	(null)	10000	10000
91 SA_REP	0.2	10000	10000
92 SA_MAN	0.2	10500	10500
93 SA_REP	0.3	10000	11000
94 PU_MAN	(null)	11000	11000
95 SA_REP	0.25	10500	11025
96 FI_MGR	(null)	12008	12008
97 AC_MGR	(null)	12008	12008
98 SA_REP	0.25	11500	12075
99 SA_MAN	0.3	11000	12100
100 SA_REP	0.3	11000	12100
101 MK_MAN	(null)	13000	13000
102 SA_MAN	0.3	12000	13200
103 SA_MAN	0.3	13500	14850
104 SA_MAN	0.4	14000	16800
105 AD_VP	(null)	17000	17000
106 AD_VP	(null)	17000	17000
107 AD_PRES	(null)	24000	24000

2. CASE 조건문 함수

```
SELECT JOB_TITLE,MIN_SALARY,  
       CASE WHEN JOB_TITLE LIKE '%Programmer' THEN MIN_SALARY*1.5  
            WHEN JOB_TITLE LIKE '%Clerk' THEN MIN_SALARY*1.3  
            WHEN JOB_TITLE LIKE '%Accountant' THEN MIN_SALARY*1.2  
            ELSE MIN_SALARY END AS REVISED_MIN_SALARY  
FROM JOBS  
WHERE JOB_TITLE LIKE '%Programmer' OR  
       JOB_TITLE LIKE '%Clerk' OR  
       JOB_TITLE LIKE '%Accountant'  
  
ORDER BY REVISED_MIN_SALARY;
```

<출력 결과>

	JOB_TITLE	MIN_SALARY	REVISED_MIN_SALARY
1	Stock Clerk	2008	2610.4
2	Purchasing Clerk	2500	3250
3	Shipping Clerk	2500	3250
4	Accountant	4200	5040
5	Public Accountant	4200	5040
6	Programmer	4000	6000

JOBS 참조 테이블에서 JOB_TITLE에 Programmer나 Clerk, Accountant 이 있는 JOB_TITLE,MIN_SALARY 를 대상으로 JOB_TITLE에 Programmer 가 들어가면 MIN_SALARY를 1.5배 올리고, Clerk 가 들어가면 MIN_SALARY를 1.3배 올리고, Accountant가 들어가면 MIN_SALARY를 1.2를 올리고 나머지는 기존 MIN_SALARY 값으로 하여 새로운 열인 REVISED_MIN_SALARY 에 출력하여 JOB_TITLE,MIN_SALARY, REVISED_MIN_SALARY를 출력하였다. 출력할 때 마지막으로 REVISED_MIN_SALARY에 대하여 오름차순으로 정렬하여 출력하였다. 총 6행이 출력되었다.

3. DECODE 함수

3.1

DECODE(열 이름,
조건1, 결과값1,
조건2, 결과값2,
조건3, 결과값3, 기본값) 새로운 열 이름

```
SELECT JOB_ID, COMMISSION_PCT,  
  
       TRUNC(COMMISSION_PCT*10) "DECODE 내용" ,  
       DECODE (TRUNC (COMMISSION_PCT*10,0),  
4, 4.00 , 3, 3.00, 2, 2.00, 1, 1.00,  
0) "수수료율"  
FROM EMPLOYEES  
WHERE JOB_ID='SA_MAN';
```

<출력 결과>

	⚡ JOB_ID	⚡ COMMISSION_PCT	⚡ DECODE 내용	⚡ 수수료율
1	SA_MAN	0.4	4	4
2	SA_MAN	0.3	3	3
3	SA_MAN	0.3	3	3
4	SA_MAN	0.3	3	3
5	SA_MAN	0.2	2	2

DECODE함수는 오라클과 같은 프로그램에서 표준 SQL은 아니지만 CASE문처럼 쓰일 수 있는 함수이다. EMPLOYEES 참조테이블에서 JOB_ID가 SA_MAN 인 조건을 만족하는 행의 JOB_ID, COMMISSION_PCT와 "DECODE 내용"의 이름의 COMMISSION_PCT에 10을 곱한 값을 버림 한 값을 가진 열, DECODE 함수를 활용 하여 "수수료율"이라는 열의 이름으로 COMMISSION_PCT에 10을 곱한 값을 1의 자리 아래는 버림 한 값이 4면 4.00으로, 3이면 3.00으로, 2라면 2.00으로, 1이라면 1.00, 기본값은 0으로 설정하여 값을 출력하였다. 총 5행이 출력되었다.

4. GROUP BY 함수

4.1

```
SELECT 그룹화할 열 이름1,  
        그룹화할 열 이름2, 집계함수  
FROM 테이블명  
WHERE 조건절  
GROUP BY 열 이름1, 열 이름2;
```

```
SELECT JOB_TITLE, MIN_SALARY, AVG(MIN_SALARY)  
FROM JOBS  
WHERE MIN_SALARY > 8000  
GROUP BY JOB_TITLE, MIN_SALARY  
ORDER BY MIN_SALARY;
```

JOBS 참조테이블에서 MIN_SALARY가 8000 초과하는 조건을 충족하는 행들에서 JOB_TITLE, MIN_SALARY로 그룹화하여 JOB_TITLE, MIN_SALARY, 그룹화된 행들의 MIN_SALARY의 평균을 AVG(MIN_SALARY) 형태로 구하여 MIN_SALARY에 오름차순으로 정렬하여 출력하였다.

* GROUP BY 절을 사용할때 GROUP BY에 명시하는 열의 경우 SELECT에서 생략해도 된다. 하지만, SELECT문에 집계함수를 쓰고 다른 열도 쓰려고 할 때 이 열이 GROUP BY에 명시되지 않았으면 오류가 발생한다. 꼭 GROUP BY에서 언급되어야 한다.

또한 WHERE 절에서는 그룹화 된 열들에 대한 조건을 설정할 수 없다. 집계함수와 같은 그룹함수를 사용할 수 없으며, 그룹에 대한 조건을 설정하고 싶을 경우 HAVING 절을 사용해야 한다.

<출력 결과>

	JOB_TITLE	MIN_SALARY	AVG(MIN_SALARY)
1	Accounting Manager	8200	8200
2	Finance Manager	8200	8200
3	Marketing Manager	9000	9000
4	Sales Manager	10000	10000
5	Administration Vice President	15000	15000
6	President	20080	20080

5. HAVING 함수

5.1

SELECT 그룹화할 열 이름1, 집계함수
FROM 테이블명
WHERE 조건절
GROUP BY 열 이름1
HAVING 열 이름1;

```
SELECT JOB_TITLE,AVG(MIN_SALARY) M_SAL
FROM JOBS
WHERE JOB_TITLE NOT LIKE '%Clerk%'
GROUP BY JOB_TITLE
HAVING AVG(MIN_SALARY)>10000
ORDER BY AVG(MIN_SALARY);
```

<출력 결과>

	JOB_TITLE	M_SAL
1	Administration Vice President	15000
2	President	20080

HAVING절은 그룹화 된 그룹에 대한 조건을 설정하기 위해 쓸 수 있다. JOBS 참조 테이블에서 JOB_TITLE에 Clerk이 포함되지 않는 조건을 만족 하는 행들에 대하여 JOB_TITLE로 그룹화하여 그룹화된 행들의 MIN_SALARY를 평균 내고 이 값이 10000을 초과할 경우 JOB_TITLE과 평균 낸 값인 AVG(MIN_SALARY)를 새로운 M_SAL 열에 출력하여 AVG(MIN_SALARY)에 대하여 오름차순으로 정렬하여 출력하였다. 총 2 행이 출력되었다.