# 프로젝트 보고서_3주차

## 0. 테이블 생성

```
CREATE TABLE EMP (
 EMP_NO VARCHAR(8) NOT NULL,
 EMP_NAME VARCHAR(20) ,
 HIREDATE DATE ,
 SALARY NUMBER(8,3) ,
 BONUS NUMBER(8,3) ,
 DEPT_CD VARCHAR(4) ,
 MANAGER VARCHAR(8) ,
 CONSTRAINT EMPLOYEE_PK
PRIMARY KEY (EMP_NO)
USING INDEX
PCTFREE 20
)
PCTFREE 10;

INSERT INTO EMP VALUES ('20063428','James' ,TO_DATE('2006-01-25', 'YYYY-MM-DD'),
9800,2400,'0000','19953472');
INSERT INTO EMP VALUES ('19953472','Owner' ,TO_DATE('1995-12-19', 'YYYY-MM-DD'),
10000,1500,'1000',NULL);
INSERT INTO EMP VALUES ('19982201','Sandra' ,TO_DATE('1998-07-07', 'YYYY-MM-DD'),
2800,2000,'1000','19953472');
INSERT INTO EMP VALUES ('20005012','Helen' ,TO_DATE('2000-12-08', 'YYYY-MM-DD'),
1000,'','1000','19982201');
INSERT INTO EMP VALUES ('20018786','David' ,TO_DATE('2001-03-01', 'YYYY-MM-DD'),
7100,100,'1000','20005012');
INSERT INTO EMP VALUES ('20032813','Nicolas' ,TO_DATE('2003-08-04', 'YYYY-MM-DD'),
7200,1100,'1000','20018786');
INSERT INTO EMP VALUES ('19963998','Bill' ,TO_DATE('1996-11-05', 'YYYY-MM-DD'),
1300,'','2000','19953472');
INSERT INTO EMP VALUES ('19972002','Paul' ,TO_DATE('1997-04-11', 'YYYY-MM-DD'),
4200,'','2000','19963998');
INSERT INTO EMP VALUES ('19976229','Fernando' ,TO_DATE('1997-05-13', 'YYYY-MM-DD'),
4300,'','2000','19972002');
INSERT INTO EMP VALUES ('19992589','John' ,TO_DATE('1999-06-20', 'YYYY-MM-DD'),
2700,400,'2000','19976229');
INSERT INTO EMP VALUES ('20027015','Karen' ,TO_DATE('2002-07-01', 'YYYY-MM-DD'),
5100,1000,'2000','19992589');
INSERT INTO EMP VALUES ('20028795','Mickey' ,TO_DATE('2002-09-01', 'YYYY-MM-DD'),
4700,2000,'2000','20027015');
INSERT INTO EMP VALUES ('19963077','Chris' ,TO_DATE('1996-03-09', 'YYYY-MM-DD'),
1400,1500,'3000','19953472');
INSERT INTO EMP VALUES ('19980185','Jane' ,TO_DATE('1998-05-03', 'YYYY-MM-DD'),
4900,'','3000','19963077');
INSERT INTO EMP VALUES ('19982915','Bob' ,TO_DATE('1998-12-29', 'YYYY-MM-DD'),
6900,'','3000','19980185');
INSERT INTO EMP VALUES ('19994601','Nick' ,TO_DATE('1999-10-17', 'YYYY-MM-DD'),
3400,'','3000','19982915');
INSERT INTO EMP VALUES ('20003969','Robert' ,TO_DATE('2000-05-12', 'YYYY-MM-DD'),
```

```
7600,'','3000','19994601');
INSERT INTO EMP VALUES ('20034532','Leonardo' ,TO_DATE('2003-09-14', 'YYYY-MM-DD'),
5600,'','3000','20003969');
INSERT INTO EMP VALUES ('20055195','Luis' ,TO_DATE('2005-04-01', 'YYYY-MM-DD'),
3200,1500,'3000','20034532');
INSERT INTO EMP VALUES ('20064224','Joy' ,TO_DATE('2006-04-25', 'YYYY-MM-DD'),
6100,'','3000','20055195');
```

# 1. 산술 연산자

## 1.1. 수행

| 수행내역 |
| --- |
| EMP 테이블 조회 |
| 산술 연산 1 |
| 산술 연산2 |
| 산술 연산3 |

## 1.2. 결과

### 1. EMP 테이블 조회

```
SELECT * FROM EMP;
```

| # | EMP_NO | EMP_NAME | HIREDATE | SALARY | BONUS | DEPT_CD | MANAGER |
|---|---|---|---|---|---|---|---|
| 1 | 19953472 | Owner | 1995/12/19 | 10,000 | 1,500 | 1000 | <NULL> |
| 2 | 19963077 | Chris | 1996/03/09 | 1,400 | 1,500 | 3000 | 19953472 |
| 3 | 19963998 | Bill | 1996/11/05 | 1,300 | <NULL> | 2000 | 19953472 |
| 4 | 19972002 | Paul | 1997/04/11 | 4,200 | <NULL> | 2000 | 19963998 |
| 5 | 19976229 | Fernando | 1997/05/13 | 4,300 | <NULL> | 2000 | 19972002 |
| 6 | 19980185 | Jane | 1998/05/03 | 4,900 | <NULL> | 3000 | 19963077 |
| 7 | 19982201 | Sandra | 1998/07/07 | 2,800 | 2,000 | 1000 | 19953472 |
| 8 | 19982915 | Bob | 1998/12/29 | 6,900 | <NULL> | 3000 | 19980185 |
| 9 | 19992589 | John | 1999/06/20 | 2,700 | 400 | 2000 | 19976229 |
| 10 | 19994601 | Nick | 1999/10/17 | 3,400 | <NULL> | 3000 | 19982915 |
| 11 | 20003969 | Robert | 2000/05/12 | 7,600 | <NULL> | 3000 | 19994601 |
| 12 | 20005012 | Helen | 2000/12/08 | 1,000 | <NULL> | 1000 | 19982201 |
| 13 | 20018786 | David | 2001/03/01 | 7,100 | 100 | 1000 | 20005012 |
| 14 | 20027015 | Karen | 2002/07/01 | 5,100 | 1,000 | 2000 | 19992589 |
| 15 | 20028795 | Mickey | 2002/09/01 | 4,700 | 2,000 | 2000 | 20027015 |
| 16 | 20032813 | Nicolas | 2003/08/04 | 7,200 | 1,100 | 1000 | 20018786 |
| 17 | 20034532 | Leonardo | 2003/09/14 | 5,600 | <NULL> | 3000 | 20003969 |
| 18 | 20055195 | Luis | 2005/04/01 | 3,200 | 1,500 | 3000 | 20034532 |
| 19 | 20063428 | James | 2006/01/25 | 9,800 | 2,400 | 0000 | 19953472 |

## 2. SALARY에 BONUS 합

```
SELECT EMP_NAME, SALARY+BONUS SAL
FROM EMP
WHERE DEPT_CD=1000;
```

| # | EMP_NAME | SAL |
|---|---|---|
| 1 | Owner | 11,500 |
| 2 | Sandra | 4,800 |
| 3 | Helen | <NULL> |
| 4 | David | 7,200 |
| 5 | Nicolas | 8,300 |

→ 산술연산자는 NULL값이 포함되면 결과도 NULL로 나온다.

→ 해결방법 : NVL함수 사용

```
SELECT EMP_NAME,NVL(SALARY+BONUS,SALARY) SAL
FROM EMP
WHERE DEPT_CD=1000;
```

| # | EMP_NAME | SAL |
|---|---|---|
| 1 | Owner | 11,500 |
| 2 | Sandra | 4,800 |
| 3 | Helen | 1,000 |
| 4 | David | 7,200 |
| 5 | Nicolas | 8,300 |

## 3. 부서번호가 1000인 직원들의 연간보상

```
SELECT EMP_NAME, SALARY, (SALARY+100)*12 "ANNUAL COMPENSATION"
FROM EMP
WHERE DEPT_CD=1000;
```

| # | EMP_NAME | SALARY | ANNUAL COMPENSATION |
|---|---|---|---|
| 1 | Owner | 10,000 | 121,200 |
| 2 | Sandra | 2,800 | 34,800 |
| 3 | Helen | 1,000 | 13,200 |
| 4 | David | 7,100 | 86,400 |
| 5 | Nicolas | 7,200 | 87,600 |

## 4. 성과금 급여

```
SELECT EMP_NAME, SALARY, ROUND(SALARY/22,2) "PERFORMACE PAY"
FROM EMP;
```

| # | EMP_NAME | SALARY | PERFORMACE PAY |
|---|---|---|---|
| 1 | Owner | 10,000 | 454.55 |
| 2 | Chris | 1,400 | 63.64 |
| 3 | Bill | 1,300 | 59.09 |
| 4 | Paul | 4,200 | 190.91 |
| 5 | Fernando | 4,300 | 195.45 |
| 6 | Jane | 4,900 | 222.73 |
| 7 | Sandra | 2,800 | 127.27 |
| 8 | Bob | 6,900 | 313.64 |
| 9 | John | 2,700 | 122.73 |
| 10 | Nick | 3,400 | 154.55 |
| 11 | Robert | 7,600 | 345.45 |
| 12 | Helen | 1,000 | 45.45 |
| 13 | David | 7,100 | 322.73 |
| 14 | Karen | 5,100 | 231.82 |
| 15 | Mickey | 4,700 | 213.64 |
| 16 | Nicolas | 7,200 | 327.27 |
| 17 | Leonardo | 5,600 | 254.55 |
| 18 | Luis | 3,200 | 145.45 |
| 19 | James | 9,800 | 445.45 |

# 2. 집합 연산자

## 2.1. 수행

| 수행내역 |
|---|
| 집합연산자 VS 일반 SELECT문 비교 |
| 집합연산자 다른 개수의 칼럼 조회 -ERROR |
| UNION 값 비교 |

## 2.2. 결과

### 1. 집합연산자 vs 일반 SELECT문

```
/*집합연산자*/
(SELECT MANAGER FROM EMP WHERE DEPT_CD=1000
MINUS
SELECT MANAGER FROM EMP WHERE DEPT_CD=2000)
```

```
UNION
(SELECT MANAGER FROM EMP WHERE DEPT_CD=2000
MINUS
SELECT MANAGER FROM EMP WHERE DEPT_CD=1000)
UNION
(SELECT MANAGER FROM EMP WHERE DEPT_CD=1000
INTERSECT
SELECT MANAGER FROM EMP WHERE DEPT_CD=2000);
```

| # | MANAGER |
|---|---|
| 1 | 19953472 |
| 2 | 19963998 |
| 3 | 19972002 |
| 4 | 19976229 |
| 5 | 19982201 |
| 6 | 19992589 |
| 7 | 20005012 |
| 8 | 20018786 |
| 9 | 20027015 |
| 10 | <NULL> |

```
SELECT DISTINCT MANAGER FROM EMP WHERE DEPT_CD IN (1000,2000) ORDER BY MANAGER;
```

| # | MANAGER |
|---|---|
| 1 | 19953472 |
| 2 | 19963998 |
| 3 | 19972002 |
| 4 | 19976229 |
| 5 | 19982201 |
| 6 | 19992589 |
| 7 | 20005012 |
| 8 | 20018786 |
| 9 | 20027015 |
| 10 | <NULL> |

→ 같은 결과를 가짐

## 2. 집합연산자 다른 개수로 조회

```
SELECT EMP_NO,MANAGER FROM EMP WHERE DEPT_CD=1000
UNION ALL
```

```
SELECT MANAGER FROM EMP WHERE DEPT_CD=2000;
```

```
[12:10:10.648]java.sql.SQLSyntaxErrorException: JDBC-8056:Invalid number of columns in query.
at line 3, column 2 of null:
SELECT MANAGER FROM EMP WHERE DEPT_CD=2000
^
```

→ 집합연산자로 SELECT문을 비교할 때 같은 개수의 칼럼으로 조회하여야함

## 3. UNION 값 비교

```
/*괄호 O*/
(SELECT MANAGER FROM EMP WHERE DEPT_CD=1000
MINUS
SELECT MANAGER FROM EMP WHERE DEPT_CD=3000)
UNION
(SELECT MANAGER FROM EMP WHERE DEPT_CD=3000
MINUS
SELECT MANAGER FROM EMP WHERE DEPT_CD=1000);

/*괄호 X*/
SELECT MANAGER FROM EMP WHERE DEPT_CD=1000
MINUS
SELECT MANAGER FROM EMP WHERE DEPT_CD=3000
UNION
SELECT MANAGER FROM EMP WHERE DEPT_CD=3000
MINUS
SELECT MANAGER FROM EMP WHERE DEPT_CD=1000;
```

| # | MANAGER |
|---|---|
| 1 | 19963077 |
| 2 | 19980185 |
| 3 | 19982201 |
| 4 | 19982915 |
| 5 | 19994601 |
| 6 | 20003969 |
| 7 | 20005012 |
| 8 | 20018786 |
| 9 | 20034532 |
| 10 | <NULL> |

| # | MANAGER |
|---|---|
| 1 | 19963077 |
| 2 | 19980185 |
| 3 | 19982915 |
| 4 | 19994601 |
| 5 | 20003969 |
| 6 | 20034532 |

→ 연산 순서가 달라짐 (결과가 다름)

- 괄호 O : MINUS(차집합1) → MINUS(차집합2) → UNION(합집합)

- 괄호 x : MINUS(차집합) → UNION(합집합) → MINUS(차집합)

# 3. 비교,논리,문자열 연산자

## 3.1. 수행

| 수행내역 |
| --- |
| 비교 연산자 |
| 논리 연산자 |
| 문자열 연산자 |
| 비교 & 논리 & 문자열 연산자 |

## 3.2. 결과

### 1. 비교연산자

```
SELECT EMP_NAME, SALARY
FROM EMP
WHERE SALARY >= 10000;
```

| # | EMP_NAME | SALARY |
| --- | --- | --- |
| 1 | Owner | 10,000 |

### 2. 논리 연산자

```
/* AND 연산자*/
SELECT EMP_NAME DEPT_CD
FROM EMP
WHERE DEPT_CD=1000 AND DEPT_CD=3000 ;
```

| # | EMP_NAME | DEPT_CD |
| --- | --- | --- |

→ 부서번호가 1000이고 3000인 사원은 존재하지 않으므로 0row 출력

```
/* OR 연산자*/
SELECT EMP_NAME DEPT_CD
FROM EMP
WHERE DEPT_CD=1000 OR DEPT_CD=3000 ;
```

| # | EMP_NAME | DEPT_CD |
|---|---|---|
| 1 | Owner | 1000 |
| 2 | Chris | 3000 |
| 3 | Jane | 3000 |
| 4 | Sandra | 1000 |
| 5 | Bob | 3000 |
| 6 | Nick | 3000 |
| 7 | Robert | 3000 |
| 8 | Helen | 1000 |
| 9 | David | 1000 |
| 10 | Nicolas | 1000 |
| 11 | Leonardo | 3000 |
| 12 | Luis | 3000 |

## 3. 문자열 연산자

```
SELECT EMP_NAME, EMP_NAME || '의 월급은' || SALARY ||'달러 입니다.'  SAL_LITERAL
FROM EMP
WHERE DEPT_CD=1000;
```

| # | EMP_NAME | SAL_LITERAL |
|---|---|---|
| 1 | Owner | Owner의 월급은10000달러 입니다. |
| 2 | Sandra | Sandra의 월급은2800달러 입니다. |
| 3 | Helen | Helen의 월급은1000달러 입니다. |
| 4 | David | David의 월급은7100달러 입니다. |
| 5 | Nicolas | Nicolas의 월급은7200달러 입니다. |

## 4. 비교 & 논리 & 문자열 연산자

```
SELECT EMP_NAME, SALARY, EMP_NAME ||'의 월급은'||SALARY
FROM EMP
WHERE SALARY >= 1000 AND SALARY <=5000;
```

| # | EMP_NAME | SALARY | LITERAL |
|---|---|---|---|
| 1 | Chris | 1,400 | Chris의 월급은1400 |
| 2 | Bill | 1,300 | Bill의 월급은1300 |
| 3 | Paul | 4,200 | Paul의 월급은4200 |
| 4 | Fernando | 4,300 | Fernando의 월급은4300 |
| 5 | Jane | 4,900 | Jane의 월급은4900 |
| 6 | Sandra | 2,800 | Sandra의 월급은2800 |
| 7 | John | 2,700 | John의 월급은2700 |
| 8 | Nick | 3,400 | Nick의 월급은3400 |
| 9 | Helen | 1,000 | Helen의 월급은1000 |
| 10 | Mickey | 4,700 | Mickey의 월급은4700 |
| 11 | Luis | 3,200 | Luis의 월급은3200 |

→ 우선순위 (비교연산자 → 논리연산자)

# 4. CASE 연산식

## 4.1. 수행

| 수행내역 |
|---|
| 일반적인 CASE 표현식 |
| ELSE 생략 후 만족하는 조건이 없으면 NULL 리턴 |
| 비교 연산자, 범위 연산자 등 사용 가능 |
| WHERE절에 사용 가능 |
| 내장 함수를 조건으로 사용 가능 |
| THEN절에서 중첩 CASE 등 추가 연산 작업 가능 |
| CASE vs DECODE |

## 4.2. 결과

### 1. 일반적인 CASE 표현식

```
SELECT EMP_NAME
,DEPT_CD
,CASE WHEN DEPT_CD=1000 THEN 'NEW YORK'
WHEN DEPT_CD=2000 THEN 'DALLAS'
ELSE 'UNKNOWN'
END AS LOC_NAME
```

```
FROM EMP
WHERE EMP_NO IN (
SELECT MANAGER
FROM EMP
);
```

| # | EMP_NAME | DEPT_CD | LOC_NAME |
|---|---|---|---|
| 1 | John | 2000 | DALLAS |
| 2 | David | 1000 | NEW YORK |
| 3 | Paul | 2000 | DALLAS |
| 4 | Robert | 3000 | UNKNOWN |
| 5 | Bill | 2000 | DALLAS |
| 6 | Leonardo | 3000 | UNKNOWN |
| 7 | Helen | 1000 | NEW YORK |
| 8 | Sandra | 1000 | NEW YORK |
| 9 | Nick | 3000 | UNKNOWN |
| 10 | Owner | 1000 | NEW YORK |
| 11 | Jane | 3000 | UNKNOWN |
| 12 | Karen | 2000 | DALLAS |
| 13 | Bob | 3000 | UNKNOWN |
| 14 | Chris | 3000 | UNKNOWN |
| 15 | Fernando | 2000 | DALLAS |

→ MANAGER를 직업을 갖는 직원 중

- DEPT_CD : 1000 → NEW YORK
- DEPT_CD : 2000 → DALLAS
- 나머지 → UNKNOWN

## 2. ELSE를 생략 후 만족하는 조건이 없으면 NULL 리턴

```
SELECT EMP_NAME
,DEPT_CD
,CASE WHEN DEPT_CD=1000 THEN 'NEW YORK'
WHEN DEPT_CD=2000 THEN 'DALLAS'
END AS LOC_NAME
FROM EMP
WHERE EMP_NO IN (
SELECT MANAGER
FROM EMP
);
```

| # | EMP_NAME | DEPT_CD | LOC_NAME |
|---|---|---|---|
| 1 | John | 2000 | DALLAS |
| 2 | David | 1000 | NEW YORK |
| 3 | Paul | 2000 | DALLAS |
| 4 | Robert | 3000 | <NULL> |
| 5 | Bill | 2000 | DALLAS |
| 6 | Leonardo | 3000 | <NULL> |
| 7 | Helen | 1000 | NEW YORK |
| 8 | Sandra | 1000 | NEW YORK |
| 9 | Nick | 3000 | <NULL> |
| 10 | Owner | 1000 | NEW YORK |
| 11 | Jane | 3000 | <NULL> |
| 12 | Karen | 2000 | DALLAS |
| 13 | Bob | 3000 | <NULL> |
| 14 | Chris | 3000 | <NULL> |
| 15 | Fernando | 2000 | DALLAS |

## 3. 비교 연산자, 범위 연산자등 사용 가능

```
SELECT EMP_NAME
,SALARY
,CASE WHEN SALARY>=8000 THEN '1등급'
WHEN SALARY>=4000 THEN '2등급'
WHEN SALARY>=1000 THEN '3등급'
END AS SAL_GRADE
FROM EMP
ORDER BY SALARY;
```

| # | EMP_NAME | SALARY | SAL_GRADE |
|---|---|---|---|
| 1 | Helen | 1,000 | 3등급 |
| 2 | Bill | 1,300 | 3등급 |
| 3 | Chris | 1,400 | 3등급 |
| 4 | John | 2,700 | 3등급 |
| 5 | Sandra | 2,800 | 3등급 |
| 6 | Luis | 3,200 | 3등급 |
| 7 | Nick | 3,400 | 3등급 |
| 8 | Paul | 4,200 | 2등급 |
| 9 | Fernando | 4,300 | 2등급 |
| 10 | Mickey | 4,700 | 2등급 |
| 11 | Jane | 4,900 | 2등급 |
| 12 | Karen | 5,100 | 2등급 |
| 13 | Leonardo | 5,600 | 2등급 |
| 14 | Bob | 6,900 | 2등급 |
| 15 | David | 7,100 | 2등급 |
| 16 | Nicolas | 7,200 | 2등급 |
| 17 | Robert | 7,600 | 2등급 |
| 18 | James | 9,800 | 1등급 |
| 19 | Owner | 10,000 | 1등급 |

## 3. WHERE절에 사용 가능

```
SELECT EMP_NAME
,SALARY
,CASE WHEN SALARY>=8000 THEN '1등급'
WHEN SALARY>=4000 THEN '2등급'
WHEN SALARY>=1000 THEN '3등급'
END AS SAL_GRADE
FROM EMP
WHERE (CASE WHEN SALARY>=8000 THEN 1
WHEN SALARY>=4000 THEN 2
WHEN SALARY>=1000 THEN 3 END)=1;
```

| # | EMP_NAME | SALARY | SAL_GRADE |
|---|---|---|---|
| 1 | Owner | 10,000 | 1등급 |
| 2 | James | 9,800 | 1등급 |

→ SAL_GRADE가 1등급인 직원

## 4. 내장 함수를 조건으로 사용 가능

```
SELECT EMP_NAME
,HIREDATE
,CASE WHEN TO_CHAR(HIREDATE,'q')='1' THEN '1분기'
WHEN TO_CHAR(HIREDATE,'q')='2' THEN '2분기'
WHEN TO_CHAR(HIREDATE,'q')='3' THEN '3분기'
WHEN TO_CHAR(HIREDATE,'q')='4' THEN '4분기'
END AS HIRE_QUARTER
FROM EMP;
```

| # | EMP_NAME | HIREDATE | HIRE_QUARTER |
|---|---|---|---|
| 1 | Owner | 1995/12/19 | 4분기 |
| 2 | Chris | 1996/03/09 | 1분기 |
| 3 | Bill | 1996/11/05 | 4분기 |
| 4 | Paul | 1997/04/11 | 2분기 |
| 5 | Fernando | 1997/05/13 | 2분기 |
| 6 | Jane | 1998/05/03 | 2분기 |
| 7 | Sandra | 1998/07/07 | 3분기 |
| 8 | Bob | 1998/12/29 | 4분기 |
| 9 | John | 1999/06/20 | 2분기 |
| 10 | Nick | 1999/10/17 | 4분기 |
| 11 | Robert | 2000/05/12 | 2분기 |
| 12 | Helen | 2000/12/08 | 4분기 |
| 13 | David | 2001/03/01 | 1분기 |
| 14 | Karen | 2002/07/01 | 3분기 |
| 15 | Mickey | 2002/09/01 | 3분기 |
| 16 | Nicolas | 2003/08/04 | 3분기 |
| 17 | Leonardo | 2003/09/14 | 3분기 |
| 18 | Luis | 2005/04/01 | 2분기 |
| 19 | James | 2006/01/25 | 1분기 |

## 5. THEN절에서 중첩 CASE 등 추가 연산 작업 가능

```
SELECT EMP_NAME
,SALARY
,DEPT_CD
,CASE WHEN DEPT_CD='1000' THEN
      CASE WHEN SALARY>=7000 THEN '1등급'
      WHEN SALARY>=3000 THEN '2등급'
      WHEN SALARY>=1000 THEN '3등급'
      END
    WHEN DEPT_CD='2000' THEN
      CASE WHEN SALARY>=8000 THEN '1등급'
      WHEN SALARY>=4000 THEN '2등급'
      WHEN SALARY>=1000 THEN '3등급'
      END
    WHEN DEPT_CD='3000' THEN
```

```
      CASE WHEN SALARY>=9000 THEN '1등급'
      WHEN SALARY>=5000 THEN '2등급'
      WHEN SALARY>=1000 THEN '3등급'
      END
    END AS SAL_GRADE
FROM EMP;
```

| # | EMP_NAME | SALARY | DEPT_CD | SAL_GRADE |
|---|---|---|---|---|
| 1 | Owner | 10,000 | 1000 | 1등급 |
| 2 | Chris | 1,400 | 3000 | 3등급 |
| 3 | Bill | 1,300 | 2000 | 3등급 |
| 4 | Paul | 4,200 | 2000 | 2등급 |
| 5 | Fernando | 4,300 | 2000 | 2등급 |
| 6 | Jane | 4,900 | 3000 | 3등급 |
| 7 | Sandra | 2,800 | 1000 | 3등급 |
| 8 | Bob | 6,900 | 3000 | 2등급 |
| 9 | John | 2,700 | 2000 | 3등급 |
| 10 | Nick | 3,400 | 3000 | 3등급 |
| 11 | Robert | 7,600 | 3000 | 2등급 |
| 12 | Helen | 1,000 | 1000 | 3등급 |
| 13 | David | 7,100 | 1000 | 1등급 |
| 14 | Karen | 5,100 | 2000 | 2등급 |
| 15 | Mickey | 4,700 | 2000 | 2등급 |
| 16 | Nicolas | 7,200 | 1000 | 1등급 |
| 17 | Leonardo | 5,600 | 3000 | 2등급 |
| 18 | Luis | 3,200 | 3000 | 3등급 |
| 19 | James | 9,800 | 0000 | <NULL> |

→ 부서번호별로 CASE중첩하여 조건문 조건 다르게 하였다.

## 6. CASE vs DECODE

```
/*CASE*/
SELECT EMP_NAME
,DEPT_CD
,CASE WHEN DEPT_CD=1000 THEN 'NEW YORK'
WHEN DEPT_CD=2000 THEN 'DALLAS'
ELSE 'NONE'
END AS LOC_NAME
FROM EMP
WHERE EMP_NO IN (
SELECT MANAGER
FROM EMP
)
ORDER BY DEPT_CD;
```

```
/*DECODE*/
SELECT EMP_NAME
,DEPT_CD
,DECODE(DEPT_CD,'1000','NEW YORK','2000','DALLAS','NONE') LOC
FROM EMP
WHERE EMP_NO IN (
SELECT MANAGER
FROM EMP
)
ORDER BY DEPT_CD;
```

| # | EMP_NAME | DEPT_CD | LOC_NAME |
|---|---|---|---|
| 1 | David | 1000 | NEW YORK |
| 2 | Helen | 1000 | NEW YORK |
| 3 | Sandra | 1000 | NEW YORK |
| 4 | Owner | 1000 | NEW YORK |
| 5 | John | 2000 | DALLAS |
| 6 | Paul | 2000 | DALLAS |
| 7 | Bill | 2000 | DALLAS |
| 8 | Karen | 2000 | DALLAS |
| 9 | Fernando | 2000 | DALLAS |
| 10 | Robert | 3000 | NONE |
| 11 | Leonardo | 3000 | NONE |
| 12 | Nick | 3000 | NONE |
| 13 | Jane | 3000 | NONE |
| 14 | Bob | 3000 | NONE |
| 15 | Chris | 3000 | NONE |

| # | EMP_NAME | DEPT_CD | LOC |
|---|---|---|---|
| 1 | David | 1000 | NEW YORK |
| 2 | Helen | 1000 | NEW YORK |
| 3 | Sandra | 1000 | NEW YORK |
| 4 | Owner | 1000 | NEW YORK |
| 5 | John | 2000 | DALLAS |
| 6 | Paul | 2000 | DALLAS |
| 7 | Bill | 2000 | DALLAS |
| 8 | Karen | 2000 | DALLAS |
| 9 | Fernando | 2000 | DALLAS |
| 10 | Robert | 3000 | NONE |
| 11 | Leonardo | 3000 | NONE |
| 12 | Nick | 3000 | NONE |
| 13 | Jane | 3000 | NONE |
| 14 | Bob | 3000 | NONE |
| 15 | Chris | 3000 | NONE |

→ 결과 같음

# 5. 변수

## 5.1. 수행

| 수행내역 |
|---|
| PL/SQL문으로 변수 선언 후 출력 |
| 구구단 출력 (4단) |

## 5.2. 결과

### 1. 변수 선언 후 출력

```
DECLARE
EX_NUM CONSTANT NUMBER := 10; --상수 선언
EX_STR VARCHAR2(10); --변수 선언

BEGIN
EX_STR := 'EXAMPLE'; --변수 초기값 설정
DBMS_OUTPUT.PUT_LINE(EX_NUM); --상수 출력
DBMS_OUTPUT.PUT_LINE(EX_STR); --변수 출력
END;
/
```

```
[22:33:41.921]10
[22:33:41.958]EXAMPLE
```

### 2. 구구단 출력

```
declare
    v_count number(10):=0;
begin
  for v_count in 1..9 loop
  dbms_output.put_line('4X'||v_count||'='||4*v_count);
  end loop;
end;
/
```

```
[22:33:43.370]4X1=4
[22:33:43.420]4X2=8
[22:33:43.467]4X3=12
[22:33:43.508]4X4=16
[22:33:43.551]4X5=20
[22:33:43.596]4X6=24
[22:33:43.637]4X7=28
[22:33:43.673]4X8=32
[22:33:43.705]4X9=36
```

# 6. BETWEEN 조건식

## 6.1. 수행

| 수행내역 |
| --- |
| BETWEEN 조건식 1 |
| BETWEEN 조건식 2 |

NOT BETWEEN 조건식

## 6.2. 결과

### 1. 입사날짜 99/01/01 과 00/12/31 사이 사원 조회

```
/*입사날짜 사이 사원 조회*/
SELECT EMP_NAME,HIREDATE
FROM EMP
WHERE HIREDATE BETWEEN '1999/01/01' AND '2000/12/31'
ORDER BY HIREDATE;
```

| # | EMP_NAME | HIREDATE |
|---|---|---|
| 1 | John | 1999/06/20 |
| 2 | Nick | 1999/10/17 |
| 3 | Robert | 2000/05/12 |
| 4 | Helen | 2000/12/08 |

### 2. 월급 3000~6000사이 사원 조회

```
SELECT EMP_NAME, SALARY
FROM EMP
WHERE SALARY NOT BETWEEN 3000 AND 6000
ORDER BY SALARY;
```

| # | EMP_NAME | SALARY |
|---|---|---|
| 1 | Helen | 1,000 |
| 2 | Bill | 1,300 |
| 3 | Chris | 1,400 |
| 4 | John | 2,700 |
| 5 | Sandra | 2,800 |
| 6 | Bob | 6,900 |
| 7 | David | 7,100 |
| 8 | Nicolas | 7,200 |
| 9 | Robert | 7,600 |
| 10 | James | 9,800 |
| 11 | Owner | 10,000 |

### 3. 사원 이름이 JAMES와 OWNER 사이가 아닌 사원 조회

```
SELECT EMP_NAME, DEPT_CD
FROM EMP
WHERE EMP_NAME NOT BETWEEN 'James' AND 'Owner'
ORDER BY EMP_NAME;
```

| # | EMP_NAME | DEPT_CD |
|---|---|---|
| 1 | Bill | 2000 |
| 2 | Bob | 3000 |
| 3 | Chris | 3000 |
| 4 | David | 1000 |
| 5 | Fernando | 2000 |
| 6 | Helen | 1000 |
| 7 | Paul | 2000 |
| 8 | Robert | 3000 |
| 9 | Sandra | 1000 |

→ 문자는 알파벳 순으로 크기가 지정된다.

# 7. IN, NOT IN 조건식

## 7.1. 수행

| 수행내역 |
|---|
| IN 조건식 1 |
| IN 조건식 2 (서브쿼리) |
| NOT IN 조건식 1 |
| NOT IN 조건식 2 (서브쿼리) |

## 7.2. 결과

### 1. 부서번호가 1000,2000인 사원 조회

```
SELECT EMP_NAME, DEPT_CD
FROM EMP
WHERE DEPT_CD IN (1000,2000);
```

| # | EMP_NAME | DEPT_CD |
|---|---|---|
| 1 | Owner | 1000 |
| 2 | Bill | 2000 |
| 3 | Paul | 2000 |
| 4 | Fernando | 2000 |
| 5 | Sandra | 1000 |
| 6 | John | 2000 |
| 7 | Helen | 1000 |
| 8 | David | 1000 |
| 9 | Karen | 2000 |
| 10 | Mickey | 2000 |
| 11 | Nicolas | 1000 |

## 2. 부서 번호가 1000,2000이 아닌 사원 조회

```
SELECT EMP_NO, EMP_NAME, DEPT_CD
FROM EMP
WHERE DEPT_CD NOT IN (1000,2000);
```

| # | EMP_NO | EMP_NAME | DEPT_CD |
|---|---|---|---|
| 1 | 19963077 | Chris | 3000 |
| 2 | 19980185 | Jane | 3000 |
| 3 | 19982915 | Bob | 3000 |
| 4 | 19994601 | Nick | 3000 |
| 5 | 20003969 | Robert | 3000 |
| 6 | 20034532 | Leonardo | 3000 |
| 7 | 20055195 | Luis | 3000 |
| 8 | 20063428 | James | 0000 |

## 3. Bob 사원과 같은 부서 번호인 사원 조회

```
SELECT EMP_NAME, DEPT_CD
FROM EMP
WHERE DEPT_CD IN (SELECT DEPT_CD FROM EMP WHERE EMP_NAME='Bob');
```

| # | EMP_NAME | DEPT_CD |
|---|---|---|
| 1 | Chris | 3000 |
| 2 | Jane | 3000 |
| 3 | Bob | 3000 |
| 4 | Nick | 3000 |
| 5 | Robert | 3000 |
| 6 | Leonardo | 3000 |
| 7 | Luis | 3000 |

### 4. David 사원과 다른 부서 번호인 사원 조회

```
SELECT EMP_NAME, DEPT_CD
FROM EMP
WHERE DEPT_CD NOT IN (SELECT DEPT_CD FROM EMP WHERE EMP_NAME='David')
ORDER BY DEPT_CD;;
```

| # | EMP_NAME | DEPT_CD |
|---|---|---|
| 1 | James | 0000 |
| 2 | Bill | 2000 |
| 3 | Paul | 2000 |
| 4 | Fernando | 2000 |
| 5 | John | 2000 |
| 6 | Karen | 2000 |
| 7 | Mickey | 2000 |
| 8 | Jane | 3000 |
| 9 | Bob | 3000 |
| 10 | Nick | 3000 |
| 11 | Robert | 3000 |
| 12 | Leonardo | 3000 |
| 13 | Luis | 3000 |
| 14 | Chris | 3000 |

# 8. EXISTS, NOT EXISTS 조건식

## 8.1. 수행

| 수행내역 |
|---|
| 서브쿼리를 쓰기 위해 TEST 사용자의 EMPLOYEE, DEPARTMENT 테이블 생성 |
| EXISTS 조건식 |
| EXISTS vs IN 조건식 (성능 비교) |

| |
|---|
| NOT EXISTS 조건식 |
| NOT EXISTS vs NOT IN 조건식 (성능 비교) |

# 8.2. 결과

## 0. EMPLOYYEE, DEPARTMENT 테이블

```
DESC TEST.EMPLOYEE;
SELECT * FROM TEST.EMPLOYEE;
```

| Column_Name | Column_Id | PK | Data_Type | Nullable | Data_Default | Comments |
|---|---|---|---|---|---|---|
| EMPNO | 0 | Y | NUMBER(4) | N | | |
| ENAME | 1 | N | VARCHAR(15) | N | | |
| JOB | 2 | N | VARCHAR(15) | Y | | |
| MANAGERNO | 3 | N | NUMBER(4) | Y | | |
| STARTDATE | 4 | N | DATE | Y | | |
| SALARY | 5 | N | NUMBER(10) | Y | | |
| DEPTNO | 6 | N | NUMBER(3) | Y | | |

| # | EMPNO | ENAME | JOB | MANAGERNO | STARTDATE | SALARY | DEPTNO |
|---|---|---|---|---|---|---|---|
| 1 | 7,369 | SMITH | CLERK | 7,902 | 0080/12/09 | 800 | 20 |
| 2 | 7,499 | ALLEN | SALESMAN | 7,839 | 0081/09/10 | 1,600 | 30 |
| 3 | 7,521 | WARD | SALESMAN | 7,698 | 0081/02/23 | 1,250 | 30 |
| 4 | 7,566 | JONES | MANAGER | 7,839 | 0081/02/04 | 2,975 | 20 |
| 5 | 7,654 | MARTIN | SALESMAN | 7,698 | 0081/02/11 | 1,250 | 30 |
| 6 | 7,698 | BLAKE | MANAGER | 7,839 | 0081/05/01 | 2,850 | 30 |
| 7 | 7,782 | CLARK | MANAGER | 7,839 | 0081/05/09 | 2,450 | 10 |
| 8 | 7,788 | SCOTT | ANALTST | 7,566 | 0082/12/22 | 3,000 | 20 |
| 9 | 7,839 | KING | PRESIDENT | <NULL> | 0081/11/17 | 5,000 | 10 |
| 10 | 7,844 | TURNER | SALESMAN | 7,698 | 0081/08/21 | 1,500 | 30 |
| 11 | 7,876 | ADAMS | CLERK | 7,788 | 0083/01/15 | 1,100 | 20 |
| 12 | 7,900 | JAMES | CLERK | 7,698 | 0081/12/11 | 950 | 30 |
| 13 | 7,902 | FORD | ANALYST | 7,566 | 0081/03/12 | 3,000 | 20 |
| 14 | 7,990 | NEW | ANALYST | 7,902 | 0081/05/12 | 2,500 | 30 |

```
DESC TEST.DEPARTMENT;
SELECT * FROM TEST.DEPARTMENT;
```

| Column_Name | Column_Id | PK | Data_Type | Nullable |
|---|---|---|---|---|
| DEPTNO | 0 | Y | NUMBER(2) | N |
| DNAME | 1 | N | VARCHAR(14) | Y |
| LOC | 2 | N | VARCHAR(13) | Y |

| # | DEPTNO | DNAME | LOC |
|---|---|---|---|
| 1 | 10 | ACCOUNTING | NEW YORK |
| 2 | 20 | RESEARCH | DALLAS |
| 3 | 30 | SALES | CHICAGO |
| 4 | 40 | OPERATIONS | BOSTON |

## 1. 부서번호가 10,20인 사원 중 월급이 3000이하인 사원 존재하는지 조회

```
SELECT * FROM TEST.DEPARTMENT D
WHERE D.DEPTNO IN (10,20)
AND EXISTS (SELECT 1 FROM TEST.EMPLOYEE E
WHERE E.SALARY <=3000 AND E.DEPTNO=D.DEPTNO);
```

| # | DEPTNO | DNAME | LOC |
|---|---|---|---|
| 1 | 10 | ACCOUNTING | NEW YORK |
| 2 | 20 | RESEARCH | DALLAS |

→ EXISTS안의 조건이 “존제” 해야만 전체 결과 출력

## 2. EXISTS 조건문 VS IN 조건문 (성능 비교)

```
/*EXISTS 조건식 */
SET TIMING ON --속도 비교
SELECT * FROM TEST.DEPARTMENT D
WHERE D.DEPTNO IN (10,20)
AND EXISTS (SELECT 1 FROM TEST.EMPLOYEE E
WHERE E.SALARY <=3000 AND E.DEPTNO=D.DEPTNO);

/*IN 조건식 */
SELECT * FROM TEST.DEPARTMENT D
WHERE D.DEPTNO IN (10,20)
AND D.DEPTNO IN (SELECT E.DEPTNO FROM TEST.EMPLOYEE E
WHERE E.SALARY <= 3000 AND E.DEPTNO=D.DEPTNO);
```

```
SQL> SET TIMING ON
SQL> SELECT * FROM TEST.DEPARTMENT D
WHERE D.DEPTNO IN (10,20)
AND D.DEPTNO IN (SELECT E.DEPTNO FROM TEST.EMPLOYEE E
WHERE E.SALARY <= 3000 AND E.DEPTNO=D.DEPTNO);   2

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS

2 rows selected.

Total elapsed time 00:00:00.004517
```

```
SQL> SET TIMING ON
SQL> SELECT * FROM TEST.DEPARTMENT D
WHERE D.DEPTNO IN (10,20)
AND D.DEPTNO IN (SELECT E.DEPTNO FROM TEST.EMPLOYEE E
WHERE E.SALARY <= 3000 AND E.DEPTNO=D.DEPTNO);   2

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS

2 rows selected.

Total elapsed time 00:00:00.000880
```

→ 속도 EXISTS 조건문 > IN 조건문

→ 항상 이런것은 아니기 때문에 데이터 종류에 따라 속도 비교해봐야 한다.

## 3. NOT EXISTS

```
SELECT * FROM TEST.DEPARTMENT D
WHERE D.DEPTNO IN (10,20)
AND NOT EXISTS (SELECT 1 FROM TEST.EMPLOYEE E
WHERE E.SALARY <=3000 AND E.DEPTNO=D.DEPTNO);
```

| # | DEPTNO | DNAME | LOC |
|---|---|---|---|

→ NOT EXISTS안의 조건이 “존재하지 않을 때” 전체 결과 출력

→ 존재하기 때문에 출력 안된다. (False)

### 4. NOT EXISTS vs NOT IN

```
/*NOT EXISTS 조건식*/
SELECT * FROM TEST.DEPARTMENT D
WHERE D.DEPTNO IN (10,20)
AND NOT EXISTS (SELECT 1 FROM TEST.EMPLOYEE E
WHERE E.SALARY <=3000 AND E.DEPTNO=D.DEPTNO);

/*NOT IN 조건식*/
SELECT * FROM TEST.DEPARTMENT D
WHERE D.DEPTNO IN (10,20)
AND D.DEPTNO NOT IN (SELECT 1 FROM TEST.EMPLOYEE E
WHERE E.SALARY <=3000 AND E.DEPTNO=D.DEPTNO);
```

| # | DEPTNO | DNAME | LOC |
|---|---|---|---|

| # | DEPTNO | DNAME | LOC |
|---|---|---|---|
| 1 | 20 | RESEARCH | DALLAS |
| 2 | 10 | ACCOUNTING | NEW YORK |

→ NOT EXISTS : 월급이 3000이하인 부서번호가 존재하여 FALSE 를 반환 → 출력값 0

→ NOT IN : 부서번호가 10,20인 사원 중 월급이 3000이하가 아닌 부서번호 반환

# 9. IS NULL,  IS NOT NULL 조건식

## 9.1. 수행

| 수행내역 |
|---|
| IS NULL 조건식 |
| IS NULL vs =NULL |
| |

| |
|---|
| IS NOT NULL |
| IS NOT NULL vs ≠NULL |

# 9.2. 결과

## 1. 보너스가 NULL인 사원 번호와 이름 조회

```
SELECT  EMP_NO,EMP_NAME,BONUS
FROM EMP
WHERE BONUS IS NULL;
```

| # | EMP_NO | EMP_NAME | BONUS |
|---|---|---|---|
| 1 | 19963998 | Bill | <NULL> |
| 2 | 19972002 | Paul | <NULL> |
| 3 | 19976229 | Fernando | <NULL> |
| 4 | 19980185 | Jane | <NULL> |
| 5 | 19982915 | Bob | <NULL> |
| 6 | 19994601 | Nick | <NULL> |
| 7 | 20003969 | Robert | <NULL> |
| 8 | 20005012 | Helen | <NULL> |
| 9 | 20034532 | Leonardo | <NULL> |
| 10 | 20064224 | Joy | <NULL> |

## 2. IS NULL vs =NULL

```
SELECT EMP_NO, EMP_NAME,BONUS
FROM EMP
WHERE BONUS =NULL;
```

| # | EMP_NO | EMP_NAME | BONUS |
|---|---|---|---|

→ 0 row

## 3. BONUS가 NULL이 아닌 사원의 이름과 BONUS를 합친 월급 조회

```
SELECT EMP_NAME, SALARY+BONUS REAL_SAL
FROM EMP
WHERE BONUS IS NOT NULL;
```

| # | EMP_NAME | REAL_SAL |
|---|---|---|
| 1 | Owner | 11,500 |
| 2 | Chris | 2,900 |
| 3 | Sandra | 4,800 |
| 4 | John | 3,100 |
| 5 | David | 7,200 |
| 6 | Karen | 6,100 |
| 7 | Mickey | 6,700 |
| 8 | Nicolas | 8,300 |
| 9 | Luis | 4,700 |
| 10 | James | 12,200 |

### 4. IS NOT NULL vs ≠NULL

```
SELECT EMP_NAME, SALARY+BONUS REAL_SAL
FROM EMP
WHERE BONUS != NULL;
```

| # | EMP_NAME | REAL_SAL |
|---|---|---|

→ 0row

# 10. LIKE,  REGEXP_LIKE 조건식

## 10.1. 수행

| 수행내역 |
|---|
| LIKE 조건식 (%) |
| LIKE 조건식 (_) |

| |
|---|
| LIKE 조건식(%,_) |
| REGEXP_LIKE 조건식 |

# 10.2. 결과

## 1. 사원 이름에 a가 두개 들어간 사원 조회

```
SELECT EMP_NAME, SALARY
FROM EMP
WHERE EMP_NAME LIKE '%a%a%';
```

| # | EMP_NAME | SALARY |
|---|---|---|
| 1 | Sandra | 2,800 |

## 2. 사원 이름에 첫번째 글자가 B이고 세글자인 사원 조회

```
SELECT EMP_NAME, SALARY
FROM EMP
WHERE EMP_NAME LIKE 'B__';
```

| # | EMP_NAME | SALARY |
|---|---|---|
| 1 | Bob | 6,900 |

## 3. 사원 이름에 뒤에서 두번째 글자가 i인 사원 조회

```
SELECT EMP_NAME, SALARY
FROM EMP
WHERE EMP_NAME LIKE '%i_';
```

| # | EMP_NAME | SALARY |
|---|---|---|
| 1 | Chris | 1,400 |
| 2 | David | 7,100 |
| 3 | Luis | 3,200 |

## 4. 사원 이름 중 B,H,S로 시작하는 사원 모두 조회

```
SELECT EMP_NAME, SALARY
FROM EMP
```

```
WHERE REGEXP_LIKE(EMP_NAME,'^B|^H|^S')
ORDER BY EMP_NAME;
```

| # | EMP_NAME | SALARY |
|---|---|---|
| 1 | Bill | 1,300 |
| 2 | Bob | 6,900 |
| 3 | Helen | 1,000 |
| 4 | Sandra | 2,800 |

# 11. 집단 함수

## 11.1. 수행

| 수행내역 |
|---|
| SELECT 절에 쓰이는 집단함수 1 |
| SELECT절에 쓰이는 집단함수 2 |
| GROUP BY 절과 함께 쓰이는 집단함수 |
| HAVING 절에 쓰이는 집단함수 |

## 11.2. 결과

### 1. 매니저 번호마다 사원의 수 조회 - COUNT(*)

```
SELECT MANAGER, COUNT(*)
FROM EMP
GROUP BY MANAGER
ORDER BY MANAGER;
```

| # | MANAGER | COUNT(*) |
|---|---|---|
| 1 | 19953472 | 4 |
| 2 | 19963077 | 1 |
| 3 | 19963998 | 1 |
| 4 | 19972002 | 1 |
| 5 | 19976229 | 1 |
| 6 | 19980185 | 1 |
| 7 | 19982201 | 1 |
| 8 | 19982915 | 1 |
| 9 | 19992589 | 1 |
| 10 | 19994601 | 1 |
| 11 | 20003969 | 1 |
| 12 | 20005012 | 1 |
| 13 | 20018786 | 1 |
| 14 | 20027015 | 1 |
| 15 | 20034532 | 1 |
| 16 | 20055195 | 1 |
| 17 | <NULL> | 1 |

→ NULL값도 포함 COUNT=1

## 2. 매니저 번호마다 사원의 수 조회 - COUNT(COLUMN명)

```
SELECT MANAGER, COUNT(MANAGER)
FROM EMP
GROUP BY MANAGER
ORDER BY MANAGER;
```

| # | MANAGER | COUNT(MAN... |
|---|---|---|
| 1 | 19953472 | 4 |
| 2 | 19963077 | 1 |
| 3 | 19963998 | 1 |
| 4 | 19972002 | 1 |
| 5 | 19976229 | 1 |
| 6 | 19980185 | 1 |
| 7 | 19982201 | 1 |
| 8 | 19982915 | 1 |
| 9 | 19992589 | 1 |
| 10 | 19994601 | 1 |
| 11 | 20003969 | 1 |
| 12 | 20005012 | 1 |
| 13 | 20018786 | 1 |
| 14 | 20027015 | 1 |
| 15 | 20034532 | 1 |
| 16 | 20055195 | 1 |
| 17 | <NULL> | 0 |

→ NULL값은 COUNT=0으로 COUNT 세지 않음

## 3. 사원의 평균 월급을 소수점 둘째짜리까지 조회

```
SELECT EMP_NAME, DEPT_CD, ROUND(AVG(SALARY),2) AVERAGE
FROM EMP
GROUP BY EMP_NAME, DEPT_CD
ORDER BY DEPT_CD;
```

| # | EMP_NAME | DEPT_CD | AVERAGE |
|---|---|---|---|
| 1 | James | 0000 | 9,800 |
| 2 | Owner | 1000 | 10,000 |
| 3 | David | 1000 | 7,100 |
| 4 | Nicolas | 1000 | 7,200 |
| 5 | Helen | 1000 | 1,000 |
| 6 | Sandra | 1000 | 2,800 |
| 7 | Paul | 2000 | 4,200 |
| 8 | Mickey | 2000 | 4,700 |
| 9 | Karen | 2000 | 5,100 |
| 10 | Bill | 2000 | 1,300 |
| 11 | John | 2000 | 2,700 |
| 12 | Fernando | 2000 | 4,300 |
| 13 | Leonardo | 3000 | 5,600 |
| 14 | Luis | 3000 | 3,200 |
| 15 | Jane | 3000 | 4,900 |
| 16 | Nick | 3000 | 3,400 |
| 17 | Chris | 3000 | 1,400 |
| 18 | Bob | 3000 | 6,900 |
| 19 | Robert | 3000 | 7,600 |
| 20 | Joy | 3000 | 6,100 |

→ GROUP BY는 SELECT문에서 집계함수를 제외한 모든 변수가 있어야한다.

### 4. 부서번호가 3번이상 조회되는 부서번호의 평균월급 조회

```
SELECT DEPT_CD, ROUND(AVG(SALARY),2) AVERAGE
FROM EMP
GROUP BY DEPT_CD
HAVING COUNT(DEPT_CD)>=3
ORDER BY DEPT_CD;
```

| # | DEPT_CD | AVERAGE |
|---|---|---|
| 1 | 1000 | 5,620 |
| 2 | 2000 | 3716.67 |
| 3 | 3000 | 4887.5 |

→ HAVING절은 GROUP BY의 조건절이라고 볼 수 있다.

# 12. 분석함수_PARTITION BY

# 12.0. 정의

- 분석함수 : 사용할 때 OVER 절을 함께 사용해야 하며, OVER절 내부에 PARTITION BY절을 사용하지 않으면 쿼리 결과 전체를 집계한다.

# 12.1. 수행

| 수행내역 |
| --- |
| SUM 집계 분석함수 |
| MAX 집계 분석함수 |
| ROW_NUMBER 순위 분석함수 |
| RANK 순위 분석함수 |
| 여러 개의 칼럼을 사용하여 그룹화 |

# 12.2. 결과

## 0. TEST 사용자가 가지고 있는 EMPLOYEE 테이블 사용

```
SELECT * FROM TEST.EMPLOYEE;
```

| # | EMPNO | ENAME | JOB | MANAGERNO | STARTDATE | SALARY | DEPTNO |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 7,200 | AMY | ANALYST | 7,902 | 0090/11/15 | <NULL> | 10 |
| 2 | 7,201 | ANNY | SALESMAN | 7,902 | 0095/11/13 | 2,000 | <NULL> |
| 3 | 7,369 | SMITH | CLERK | 7,902 | 0080/12/09 | 800 | 20 |
| 4 | 7,499 | ALLEN | SALESMAN | 7,839 | 0081/09/10 | 1,600 | 30 |
| 5 | 7,521 | WARD | SALESMAN | 7,698 | 0081/02/23 | 1,250 | 30 |
| 6 | 7,566 | JONES | MANAGER | 7,839 | 0081/02/04 | 2,975 | 20 |
| 7 | 7,654 | MARTIN | SALESMAN | 7,698 | 0081/02/11 | 1,250 | 30 |
| 8 | 7,698 | BLAKE | MANAGER | 7,839 | 0081/05/01 | 2,850 | 30 |
| 9 | 7,782 | CLARK | MANAGER | 7,839 | 0081/05/09 | 2,450 | 10 |
| 10 | 7,788 | SCOTT | ANALTST | 7,566 | 0082/12/22 | 3,000 | 20 |
| 11 | 7,839 | KING | PRESIDENT | <NULL> | 0081/11/17 | 5,000 | 10 |
| 12 | 7,844 | TURNER | SALESMAN | 7,698 | 0081/08/21 | 1,500 | 30 |
| 13 | 7,876 | ADAMS | CLERK | 7,788 | 0083/01/15 | 1,100 | 20 |
| 14 | 7,900 | JAMES | CLERK | 7,698 | 0081/12/11 | 950 | 30 |
| 15 | 7,902 | FORD | ANALYST | 7,566 | 0081/03/12 | 3,000 | 20 |
| 16 | 7,990 | NEW | ANALYST | 7,902 | 0081/05/12 | 2,500 | 30 |

## 1. 직업이 MANAGER, SALESMAN인 사원의 직군별 월급 합 조회

```
SELECT ENAME,JOB,SALARY,SUM(SALARY) OVER(PARTITION BY JOB) PART
FROM TEST.EMPLOYEE
```

```
WHERE JOB IN ('MANAGER','SALESMAN')
ORDER BY JOB;
```

| # | ENAME | JOB | SALARY | PART |
|---|---|---|---|---|
| 1 | JONES | MANAGER | 2,975 | 8,275 |
| 2 | BLAKE | MANAGER | 2,850 | 8,275 |
| 3 | CLARK | MANAGER | 2,450 | 8,275 |
| 4 | ALLEN | SALESMAN | 1,600 | 7,600 |
| 5 | WARD | SALESMAN | 1,250 | 7,600 |
| 6 | ANNY | SALESMAN | 2,000 | 7,600 |
| 7 | MARTIN | SALESMAN | 1,250 | 7,600 |
| 8 | TURNER | SALESMAN | 1,500 | 7,600 |

## 2. 직업이 MANAGER, SALESMAN인 사원의 직군별 월급의 최댓값 조회

```
SELECT ENAME,JOB,SALARY,MAX(SALARY) OVER(PARTITION BY JOB) PART
FROM TEST.EMPLOYEE
WHERE JOB IN ('MANAGER','SALESMAN')
ORDER BY SALARY;
```

| # | ENAME | JOB | SALARY | PART |
|---|---|---|---|---|
| 1 | WARD | SALESMAN | 1,250 | 2,000 |
| 2 | MARTIN | SALESMAN | 1,250 | 2,000 |
| 3 | TURNER | SALESMAN | 1,500 | 2,000 |
| 4 | ALLEN | SALESMAN | 1,600 | 2,000 |
| 5 | ANNY | SALESMAN | 2,000 | 2,000 |
| 6 | CLARK | MANAGER | 2,450 | 2,975 |
| 7 | BLAKE | MANAGER | 2,850 | 2,975 |
| 8 | JONES | MANAGER | 2,975 | 2,975 |

## 3. 직군별 월급의 순서를 오름차순으로 조회 (ROW_NUMBER())

```
SELECT ENAME,JOB,SALARY,ROW_NUMBER() OVER(PARTITION BY JOB ORDER BY SALARY) AS RN
FROM TEST.EMPLOYEE
WHERE JOB IN ('MANAGER','SALESMAN')
ORDER BY SALARY;
```

| # | ENAME | JOB | SALARY | RN |
|---|---|---|---|---|
| 1 | MARTIN | SALESMAN | 1,250 | 1 |
| 2 | WARD | SALESMAN | 1,250 | 2 |
| 3 | TURNER | SALESMAN | 1,500 | 3 |
| 4 | ALLEN | SALESMAN | 1,600 | 4 |
| 5 | ANNY | SALESMAN | 2,000 | 5 |
| 6 | CLARK | MANAGER | 2,450 | 1 |
| 7 | BLAKE | MANAGER | 2,850 | 2 |
| 8 | JONES | MANAGER | 2,975 | 3 |

→ OVER절 안에 있는 ORDER BY를 해주어 순위 매겨준다.

## 4. 직군별 월급의 순서를 오름차순으로 조회 (RANK())

```
SELECT ENAME,JOB,SALARY,RANK() OVER(PARTITION BY JOB ORDER BY SALARY) AS RN
FROM TEST.EMPLOYEE
WHERE JOB IN ('MANAGER','SALESMAN')
ORDER BY SALARY;
```

| # | ENAME | JOB | SALARY | RN |
|---|---|---|---|---|
| 1 | WARD | SALESMAN | 1,250 | 1 |
| 2 | MARTIN | SALESMAN | 1,250 | 1 |
| 3 | TURNER | SALESMAN | 1,500 | 3 |
| 4 | ALLEN | SALESMAN | 1,600 | 4 |
| 5 | ANNY | SALESMAN | 2,000 | 5 |
| 6 | CLARK | MANAGER | 2,450 | 1 |
| 7 | BLAKE | MANAGER | 2,850 | 2 |
| 8 | JONES | MANAGER | 2,975 | 3 |

→ 직군별 사원중 같은 월급을 가지면 같은 순위로 나온다.

```
/*ORDER BY에 ENAME 추가*/
SELECT ENAME,JOB,SALARY,RANK() OVER(PARTITION BY JOB ORDER BY SALARY,ENAME) AS RN
FROM TEST.EMPLOYEE
WHERE JOB IN ('MANAGER','SALESMAN')
ORDER BY SALARY;
```

| # | ENAME | JOB | SALARY | RN |
|---|---|---|---|---|
| 1 | MARTIN | SALESMAN | 1,250 | 1 |
| 2 | WARD | SALESMAN | 1,250 | 2 |
| 3 | TURNER | SALESMAN | 1,500 | 3 |
| 4 | ALLEN | SALESMAN | 1,600 | 4 |
| 5 | ANNY | SALESMAN | 2,000 | 5 |
| 6 | CLARK | MANAGER | 2,450 | 1 |
| 7 | BLAKE | MANAGER | 2,850 | 2 |
| 8 | JONES | MANAGER | 2,975 | 3 |

→ 같은 월급을 가진 사원에게 다른 순위를 부여하기 위해 ORDER BY에 ENAME 칼럼도 추가

### 5. 여러 개의 칼럼을 사용하여 그룹화

```
SELECT EMPNO,JOB,DEPTNO,SALARY
,SUM(SALARY) OVER (PARTITION BY JOB, DEPTNO)
FROM TEST.EMPLOYEE
WHERE JOB IN ('MANAGER','SALESMAN')
ORDER BY SALARY;
```

| # | EMPNO | JOB | DEPTNO | SALARY | PART |
|---|---|---|---|---|---|
| 1 | 7,521 | SALESMAN | 30 | 1,250 | 5,600 |
| 2 | 7,654 | SALESMAN | 30 | 1,250 | 5,600 |
| 3 | 7,844 | SALESMAN | 30 | 1,500 | 5,600 |
| 4 | 7,499 | SALESMAN | 30 | 1,600 | 5,600 |
| 5 | 7,201 | SALESMAN | <NULL> | 2,000 | 2,000 |
| 6 | 7,782 | MANAGER | 10 | 2,450 | 2,450 |
| 7 | 7,698 | MANAGER | 30 | 2,850 | 2,850 |
| 8 | 7,566 | MANAGER | 20 | 2,975 | 2,975 |

→ 직군별, 월급별로 월급의 합 PART칼럼에 추가

# 13. 문자함수_1

# 13.1. 수행

| 수행내역 |
| --- |
| 2개의 문자값 결합 |
| 첫번째 문자를 대문자로 변환 |
| 나머지 공간을 지정한 문자로 채우기 |
| 정의된 문장의 왼쪽부터 지정된 단어가 발견되면 제거 |

# 13.2. 결과

## 1. 2개의 문자값 결합

```
SELECT CONCAT(CONCAT(ENAME,' IS A '),JOB) LITERAL
FROM TEST.EMPLOYEE;
```

| # | LITERAL |
| --- | --- |
| 1 | AMY IS A ANALYST |
| 2 | ANNY IS A SALESMAN |
| 3 | SMITH IS A CLERK |
| 4 | ALLEN IS A SALESMAN |
| 5 | WARD IS A SALESMAN |
| 6 | JONES IS A MANAGER |
| 7 | MARTIN IS A SALESMAN |
| 8 | BLAKE IS A MANAGER |
| 9 | CLARK IS A MANAGER |
| 10 | SCOTT IS A ANALTST |
| 11 | KING IS A PRESIDENT |
| 12 | TURNER IS A SALESMAN |
| 13 | ADAMS IS A CLERK |
| 14 | JAMES IS A CLERK |
| 15 | FORD IS A ANALYST |
| 16 | NEW IS A ANALYST |

## 2. 첫번째 문자를 대문자로 변환

```
SELECT INITCAP('hi my name is jeongmin') INITCAP
FROM DUAL;
```

| # | INITCAP |
|---|---|
| 1 | Hi My Name Is Jeongmin |

### 3. 나머지 공간을 지정한 문자로 채우기

```
SELECT LPAD('PAGE1',15,'*') PAGE FROM DUAL;
```

| # | PAGE |
|---|---|
| 1 | **********PAGE1 |

→ 오른쪽부터 문자로 채우기

### 4. 정의된 문장의 왼쪽부터 지정된 단어가 발견되면 제거

```
SELECT LTRIM('xyxXxyLAST WORD','xy') LTRIM
FROM DUAL;
```

| # | LTRIM |
|---|---|
| 1 | XxyLAST WORD |

# 14. 문자함수_2

## 14.1. 수행

| 수행내역 |
|---|
| 정의된 문장에서 해당 문자가 발견되면 지정된 문자로 변경 |
| 정의된 문자의 오른쪽 나머지 공간을 지정한 문자로 채우기 |
| 정의된 문자의 오른쪽부터 지정된 단어가 발견되면 제거 |
| 정의된 문장의 지정된 위치부터 해당 길이 만큼만 추출 |

## 14.2. 결과

### 1. 정의된 문장에서 해당 문자 발견되면 지정된 문자로 변경

```
SELECT REPLACE('JACK and JUE','J','BL') REPLACE
FROM DUAL;
```

### 2. 정의된 문자의 오른쪽 나머지 공간을 지정한 문자로 채우기

```
/*RPAD(칼럼명,문자열 크기,채울 문자)*/
SELECT RPAD(EMP_NAME,11,'AB') RPAD
FROM EMP
WHERE EMP_NAME='Jane';
```

### 3. 정의된 문자의 오른쪽부터 지정된 단어가 발견되면 제거

```
SELECT RTRIM('JaneyxXxy','yxXxy') RTRIM
FROM DUAL;
```

| # | RTRIM |
|---|---|
| 1 | Jane |

### 4. 정의된 문장의 지정된 위치부터 해당 길이 만큼만 추출

```
/*SUBSTR(문자열,시작위치,추출할 길이*/
SELECT SUBSTR('ABCDEFG',3,2) SUBSTR FROM DUAL;
```

| # | SUBSTR |
|---|---|
| 1 | CD |

# 15. 문자함수_3

## 15.1. 수행

| 수행내역 |
|---|
| 정의된 문장의 뒤에서부터 지정된 위치의 해당 길이 만큼만 추출 |
| 문자 'Q'를 ASCII 코드로 변환 |
| 정의된 문장에서 지정된 위치에 존재하는 문자의 위치 값을 찾아줌 |
| 정의된 문장의 길이를 변환 |

## 15.2. 결과

### 1. 정의된 문장의 뒤에서부터 지정된 위치의 해당 길이 만큼만 추출

```
SELECT SUBSTR('ABCDEFG',-3,2) "-SUBSTR" FROM DUAL;
```

| # | -SUBSTR |
|---|---|
| 1 | EF |

→ 뒤에서 3번째 문자 중 두 글자 추출

### 2. 문자 'Q'를 ASCII 코드로 변환

```
SELECT ASCII('Q') FROM DUAL;
```

### 3. 정의된 문장에서 지정된 위치에 존재하는 문자의 위치 값을 찾아줌

```
SELECT INSTR('CORPORATE FLOOR','OR',3,2) INSTR FROM DUAL;
```

| # | INSTR |
|---|---|
| 1 | 14 |

→ 문자열에서 3번째 문자를 기준으로 2번째로 OR을 찾을 수 있는 위치 값

### 4. 정의된 문장의 길이를 변환

```
SELECT LENGTHB('정MIN0103') "LENGTH(BYTE)" FROM DUAL;
```

| # | LENGTH(BYTE) |
|---|---|
| 1 | 10 |

→ 3BYTE+1BYTE*3+1BYTE*4=10

# 16. GREATEST 함수

## 16.1. 수행

| 수행내역 |
| --- |
| 숫자 비교 |
| 문자 비교 |
| 날짜 비교 |
| NULL값 포함하여 비교 |
| 숫자와 문자 같이 비교 |

# 16.2. 결과

## 1. 숫자 비교

```
SELECT GREATEST(100, 200, 300, 400, 500) "GREATEST_1"
FROM DUAL;
```

| # | GREATEST_1 |
| --- | --- |
| 1 | 500 |

## 2. 문자 비교

```
SELECT GREATEST('AAA','BBB','CCC','DDD') "GREATEST_2"
FROM DUAL;
```

| # | GREATEST_2 |
| --- | --- |
| 1 | DDD |

→ 알파벳순으로 뒤로 갈수록 크기가 크다

## 3. 날짜 비교

```
SELECT SYSDATE, GREATEST(SYSDATE,SYSDATE+1,SYSDATE+2) "GREATEST_3"
FROM DUAL;
```

| # | SYSDATE | GREATEST_3 |
| --- | --- | --- |
| 1 | 2022/11/18 | 2022/11/20 |

### 4. NULL값 포함하여 비교

```
SELECT GREATEST(100,200,300,400,NULL) NULL_YES
FROM DUAL;
```

| # | NULL_YES |
|---|---|
| 1 | <NULL> |

→ NULL값을 포함하여 비교하면 결과가 무조건 NULL값

### 5. 숫자와 문자 함께 비교 - ERROR

```
SELECT GREATEST(100,200,300,400,'AAA') ERROR
FROM DUAL;
```

> [11:36:50.119]java.sql.SQLException: JDBC-5074:Given string does not represent a number in proper format.

# 17. NVL 함수

## 17.1. 수행

| 수행내역 |
|---|
| 지정된 값으로 NULL값 대체 |
| NULL 값 포함하여 산술연산자 → NVL()로 대체 |
| NULL 값 포함하여 집계함수 → NVL()로 대체 X |

## 17.2. 결과

### 1. BONUS (NULL→0), EMP_NAME (NULL→*) 로 대체

```
SELECT NVL(BONUS,0) "NVL_BONUS", NVL(EMP_NAME,'*') "NVL_NAME"
FROM EMP;
```

### 2. NULL값 포함하여 산술연산자

```
SELECT SALARY+BONUS "NULL_SARARY",NVL(SALARY+BONUS, 0) "REAL_SALARY"
FROM EMP;
```

| # | NULL_SARARY | REAL_SALARY |
|---|---|---|
| 1 | 11,500 | 11,500 |
| 2 | 2,900 | 2,900 |
| 3 | <NULL> | 0 |
| 4 | <NULL> | 0 |
| 5 | <NULL> | 0 |
| 6 | <NULL> | 0 |
| 7 | 4,800 | 4,800 |
| 8 | <NULL> | 0 |
| 9 | 3,100 | 3,100 |
| 10 | <NULL> | 0 |
| 11 | <NULL> | 0 |
| 12 | <NULL> | 0 |
| 13 | 7,200 | 7,200 |
| 14 | 6,100 | 6,100 |
| 15 | 6,700 | 6,700 |
| 16 | 8,300 | 8,300 |
| 17 | <NULL> | 0 |
| 18 | 4,700 | 4,700 |
| 19 | 12,200 | 12,200 |
| 20 | <NULL> | 0 |

## 3. NULL 값 포함하여 집계함수

```
SELECT DEPT_CD, BONUS
FROM EMP
ORDER BY DEPT_CD;

SELECT DEPT_CD, SUM(BONUS)
FROM EMP
GROUP BY DEPT_CD
ORDER BY DEPT_CD;
```

| # | DEPT_CD | SUM(BONUS) |
|---|---|---|
| 1 | 0000 | 2,400 |
| 2 | 1000 | 4,700 |
| 3 | 2000 | 3,400 |
| 4 | 3000 | 3,000 |

| # | DEPT_CD | BONUS |
|---|---|---|
| 1 | 0000 | 2,400 |
| 2 | 1000 | 1,500 |
| 3 | 1000 | 2,000 |
| 4 | 1000 | 1,100 |
| 5 | 1000 | <NULL> |
| 6 | 1000 | 100 |
| 7 | 2000 | <NULL> |
| 8 | 2000 | <NULL> |
| 9 | 2000 | <NULL> |
| 10 | 2000 | 400 |
| 11 | 2000 | 2,000 |
| 12 | 2000 | 1,000 |
| 13 | 3000 | 1,500 |
| 14 | 3000 | <NULL> |
| 15 | 3000 | <NULL> |
| 16 | 3000 | <NULL> |
| 17 | 3000 | <NULL> |
| 18 | 3000 | <NULL> |
| 19 | 3000 | 1,500 |
| 20 | 3000 | <NULL> |

→ 집계함수는 NULL값 0으로 받아들이기 때문에 NVL함수로 대체하지 않아도 됨.

# 18. 숫자함수_1

## 18.1. 수행

| 수행내역 |
|---|
| 절대값으로 변환 |
| 정의된 값의 올림된 값으로 변환 |
| 정의된 값의 내림된 값으로 변환 |
| 정의된 산술식의 COSINE 값으로 소수 둘째짜리까지 변환 |

## 18.2. 결과

### 1. 절대값으로 변환

```
SELECT EMP_NAME,NVL(BONUS-SALARY,0) NVL,ABS(NVL(BONUS-SALARY,0)) ABS
FROM EMP
ORDER BY 2;
```

| # | EMP_NAME | NVL | ABS |
|---|---|---|---|
| 1 | Owner | -8,500 | 8,500 |
| 2 | James | -7,400 | 7,400 |
| 3 | David | -7,000 | 7,000 |
| 4 | Nicolas | -6,100 | 6,100 |
| 5 | Karen | -4,100 | 4,100 |
| 6 | Mickey | -2,700 | 2,700 |
| 7 | John | -2,300 | 2,300 |
| 8 | Luis | -1,700 | 1,700 |
| 9 | Sandra | -800 | 800 |
| 10 | Leonardo | 0 | 0 |
| 11 | Helen | 0 | 0 |
| 12 | Robert | 0 | 0 |
| 13 | Nick | 0 | 0 |
| 14 | Bob | 0 | 0 |
| 15 | Jane | 0 | 0 |
| 16 | Fernando | 0 | 0 |
| 17 | Paul | 0 | 0 |
| 18 | Bill | 0 | 0 |
| 19 | Joy | 0 | 0 |
| 20 | Chris | 100 | 100 |

## 2. 정의된 값의 올림된 값으로 변환

```
SELECT CEIL(15.789878798470309803) CEIL FROM DUAL;
```

| # | CEIL |
|---|---|
| 1 | 16 |

## 3. 정의된 값의 내림된 값으로 변환

```
SELECT FLOOR(15.789878798470309803) FLOOR FROM DUAL;
```

| # | FLOOR |
|---|---|
| 1 | 15 |

## 4. 정의된 산술식의 COSINE 값으로 소수 둘째짜리까지 변환

```
SELECT ROUND(COS(60*3.14/180),2) FROM DUAL;
```

| # | ROUND(COS... |
|---|---|
| 1 | 0.5 |

→ COS(60도)=1/2

# 19. 숫자함수_2

## 19.1. 수행

| 수행내역 |
|---|
| 지수승값 계산 |
| 정의된 수로 앞에 정의된 수를 나눈 나머지 값 반환 |
| 음수이면 -1,0이면0,양수이면1 반환 |
| 정의된 수를 지정한 자릿수에서 절삭 |

## 19.2. 결과

### 1. 지수승값 계산

```
SELECT EXP(4) EXP FROM DUAL;
```

| # | EXP |
|---|---|
| 1 | 54.598150... |

→ 자연로그 e의 4승

### 2. 정의된 수로 앞에 정의된 수를 나눈 나머지값 반환

```
SELECT MOD(11,4) FROM DUAL;
```

| # | MOD(11,4) |
|---|---|
| 1 | 3 |

→ 11/4=2…3 이므로 3출력

## 3. 음수면 -1, 0이면 0, 양수이면 1 반환

```
SELECT NVL(BONUS-SALARY,0) NVL, SIGN(NVL(BONUS-SALARY,0)) SIGN
FROM EMP;
```

| # | NVL | SIGN |
|---|---|---|
| 1 | -8,500 | -1 |
| 2 | 100 | 1 |
| 3 | 0 | 0 |
| 4 | 0 | 0 |
| 5 | 0 | 0 |
| 6 | 0 | 0 |
| 7 | -800 | -1 |
| 8 | 0 | 0 |
| 9 | -2,300 | -1 |
| 10 | 0 | 0 |
| 11 | 0 | 0 |
| 12 | 0 | 0 |
| 13 | -7,000 | -1 |
| 14 | -4,100 | -1 |
| 15 | -2,700 | -1 |
| 16 | -6,100 | -1 |
| 17 | 0 | 0 |
| 18 | -1,700 | -1 |
| 19 | -7,400 | -1 |
| 20 | 0 | 0 |

## 4. 정의된 수를 지정한 자릿수에서 절삭

```
SELECT TRUNC(15.9876, 2) FROM DUAL;
SELECT TRUNC(15.9876, -1) FROM DUAL;
```

| # | TRUNC(15.... |
|---|---|
| 1 | 15.98 |

→ 소수 둘째자리까지 조회하고 그 뒤 절삭

| # | TRUNC(15.... |
|---|---|
| 1 | 10 |

→ 일의자리까지 절삭

# 20. 날짜함수

## 20.1. 수행

| 수행내역 |
| --- |
| 현재 시스템 날짜 반환 |
| 해당 날짜에 지정한 달 수 만큼 더하기 |
| 정의된 날짜의 달에서 마지막 일이 몇일인지 구하기 |
| 정의된 두 날짜간의 차이 값 |

## 20.2. 결과

### 1. 현재 시스템 날짜 반환

```
SELECT SYSDATE FROM DUAL;
SELECT SYSTIMESTAMP FROM DUAL; --시간까지 출력
```

| # | SYSDATE |
| --- | --- |
| 1 | 2022/11/18 |

| # | SYSTIMESTAMP |
| --- | --- |
| 1 | 2022-11-18 15:19:16.956518 |

### 2. 해당 날짜에 지정한 달 수만큼 더하기

```
SELECT HIREDATE, ADD_MONTHS(HIREDATE,1)
FROM EMP WHERE DEPT_CD=1000;
```

| # | HIREDATE | ADD_MONTH... |
| --- | --- | --- |
| 1 | 1995/12/19 | 1996/01/19 |
| 2 | 1998/07/07 | 1998/08/07 |
| 3 | 2000/12/08 | 2001/01/08 |
| 4 | 2001/03/01 | 2001/04/01 |
| 5 | 2003/08/04 | 2003/09/04 |

### 3. 정의된 날짜의 달에서 마지막 일이 몇일인지 구하기

```
SELECT HIREDATE, LAST_DAY(HIREDATE)
FROM EMP WHERE DEPT_CD=2000;
```

| # | HIREDATE | LAST_DAY(... |
|---|---|---|
| 1 | 1996/11/05 | 1996/11/30 |
| 2 | 1997/04/11 | 1997/04/30 |
| 3 | 1997/05/13 | 1997/05/31 |
| 4 | 1999/06/20 | 1999/06/30 |
| 5 | 2002/07/01 | 2002/07/31 |
| 6 | 2002/09/01 | 2002/09/30 |

### 4. 정의된 두 날짜간의 차이 값 (SYSDATE-HIREDATE)

```
SELECT HIREDATE, MONTHS_BETWEEN(SYSDATE,HIREDATE)
FROM EMP WHERE DEPT_CD=3000;
```

| # | HIREDATE | MONTHS_BE... |
|---|---|---|
| 1 | 1996/03/09 | 320.31097... |
| 2 | 1998/05/03 | 294.50452... |
| 3 | 1998/12/29 | 286.66581... |
| 4 | 1999/10/17 | 277.05291... |
| 5 | 2000/05/12 | 270.21420... |
| 6 | 2003/09/14 | 230.14968... |
| 7 | 2005/04/01 | 211.56904... |
| 8 | 2006/04/25 | 198.79484... |

# 21. 변환함수로 포맷 변경하기_1

## 21.1. 수행

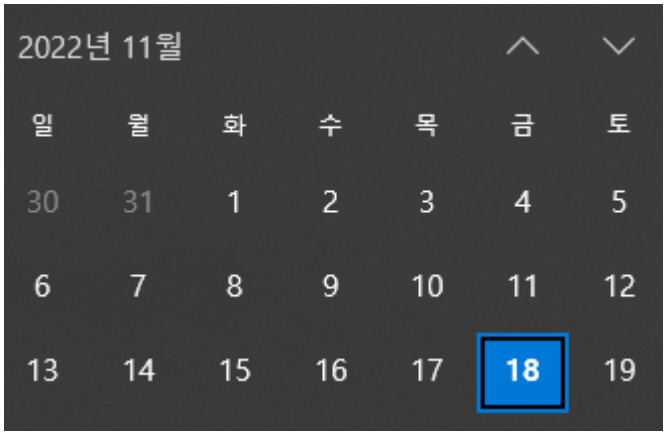
| 수행내역 |
|---|
| 현재 날짜가 한 주에서 몇번째 일 |
| - 정의된 날짜의 출력 포맷 DD-MM-YY로 출력 - DD-MM-YY로 출력할 때 맨앞 0값 제거 |
| 문자형 숫자형으로 변환 |
| 숫자형 문자형으로 변환 |
| 현재 시간을 AM, PM 표기법으로 출력 |

## 21.2. 결과

### 1. 현재 날짜가 한 주에서 몇번째 일 → 6번째

```
SELECT SYSDATE,TO_CHAR(SYSDATE,'D') WEEK_DATE FROM DUAL;
```

| # | SYSDATE | WEEK_DATE |
|---|---|---|
| 1 | 2022/11/18 | 6 |




### 2. 정의된 날짜의 출력 포맷 DD-MM-YY로 출력

```
SELECT EMP_NAME,HIREDATE,TO_CHAR(HIREDATE,'DD-MM-YY') HIREDATE FROM EMP;
```

| # | EMP_NAME | HIREDATE | HIREDATE |
|---|---|---|---|
| 1 | Owner | 1995/12/19 | 19-12-95 |
| 2 | Chris | 1996/03/09 | 09-03-96 |
| 3 | Bill | 1996/11/05 | 05-11-96 |
| 4 | Paul | 1997/04/11 | 11-04-97 |
| 5 | Fernando | 1997/05/13 | 13-05-97 |
| 6 | Jane | 1998/05/03 | 03-05-98 |
| 7 | Sandra | 1998/07/07 | 07-07-98 |
| 8 | Bob | 1998/12/29 | 29-12-98 |
| 9 | John | 1999/06/20 | 20-06-99 |
| 10 | Nick | 1999/10/17 | 17-10-99 |
| 11 | Robert | 2000/05/12 | 12-05-00 |
| 12 | Helen | 2000/12/08 | 08-12-00 |
| 13 | David | 2001/03/01 | 01-03-01 |
| 14 | Karen | 2002/07/01 | 01-07-02 |
| 15 | Mickey | 2002/09/01 | 01-09-02 |
| 16 | Nicolas | 2003/08/04 | 04-08-03 |
| 17 | Leonardo | 2003/09/14 | 14-09-03 |
| 18 | Luis | 2005/04/01 | 01-04-05 |
| 19 | James | 2006/01/25 | 25-01-06 |
| 20 | Joy | 2006/04/25 | 25-04-06 |

```
SELECT EMP_NAME,HIREDATE, TO_CHAR(HIREDATE,'fmDD-MM-YY') HIREDATE FROM EMP;
```

| # | EMP_NAME | HIREDATE | HIREDATE |
|---|---|---|---|
| 1 | Owner | 1995/12/19 | 19-12-95 |
| 2 | Chris | 1996/03/09 | 9-3-96 |
| 3 | Bill | 1996/11/05 | 5-11-96 |
| 4 | Paul | 1997/04/11 | 11-4-97 |
| 5 | Fernando | 1997/05/13 | 13-5-97 |
| 6 | Jane | 1998/05/03 | 3-5-98 |
| 7 | Sandra | 1998/07/07 | 7-7-98 |
| 8 | Bob | 1998/12/29 | 29-12-98 |
| 9 | John | 1999/06/20 | 20-6-99 |
| 10 | Nick | 1999/10/17 | 17-10-99 |
| 11 | Robert | 2000/05/12 | 12-5-0 |
| 12 | Helen | 2000/12/08 | 8-12-0 |
| 13 | David | 2001/03/01 | 1-3-1 |
| 14 | Karen | 2002/07/01 | 1-7-2 |
| 15 | Mickey | 2002/09/01 | 1-9-2 |
| 16 | Nicolas | 2003/08/04 | 4-8-3 |
| 17 | Leonardo | 2003/09/14 | 14-9-3 |
| 18 | Luis | 2005/04/01 | 1-4-5 |
| 19 | James | 2006/01/25 | 25-1-6 |
| 20 | Joy | 2006/04/25 | 25-4-6 |

## 3. 문자형 숫자형으로 변환

```
SELECT TO_NUMBER('100') FROM DUAL;
```

| # | TO_NUMBER... |
|---|---|
| 1 | 100 |

```
SELECT TO_NUMBER('JEONGMIN') FROM DUAL;
```

> [16:05:23.375]java.sql.SQLException: JDBC-5074:Given string does not represent a number in proper format.

→ 알파벳으로 되어있는 문자형은 TO_NUMBER 형으로 숫자형으로 변활할 수 없음

## 4. 숫자형 문자형으로 변환

```
SELECT TO_CHAR(100) FROM DUAL;
```

| # | TO_CHAR(100) |
|---|---|
| 1 | 100 |

### 5. 현재 시간을 AM, PM 표기법으로 출력

```
SELECT TO_CHAR(SYSDATE,'PM HH:MI') FROM DUAL;
```

| # | TO_CHAR(S... |
|---|---|
| 1 | 오후 04:07 |

# 22. 변환함수로 포맷 변경하기_2

## 22.1. 수행

| 수행내역 |
|---|
| 숫자값 출력할 때 금액표시 |
| 정의된 날짜를 지정한 포맷(TO_DATE( )) -ERROR |
| 정의된 날짜를 지정한 포맷 (TO_DATE( )) |
| 정의된 날짜를 지정한 포맷 (TO_CHAR( )) |

## 22.2. 결과

### 1. 숫자값 출력할 때 금액 표시

```
SELECT SALARY,TO_CHAR(SALARY,'$909,999') FROM EMP;
```

| # | SALARY | TO_CHAR(S... |
|---|---|---|
| 1 | 10,000 | $10,000 |
| 2 | 1,400 | $01,400 |
| 3 | 1,300 | $01,300 |
| 4 | 4,200 | $04,200 |
| 5 | 4,300 | $04,300 |
| 6 | 4,900 | $04,900 |
| 7 | 2,800 | $02,800 |
| 8 | 6,900 | $06,900 |
| 9 | 2,700 | $02,700 |
| 10 | 3,400 | $03,400 |
| 11 | 7,600 | $07,600 |
| 12 | 1,000 | $01,000 |
| 13 | 7,100 | $07,100 |
| 14 | 5,100 | $05,100 |
| 15 | 4,700 | $04,700 |
| 16 | 7,200 | $07,200 |
| 17 | 5,600 | $05,600 |
| 18 | 3,200 | $03,200 |
| 19 | 9,800 | $09,800 |
| 20 | 6,100 | $06,100 |

## 2. 정의된 날짜를 지정(1) - ERROR

```
SELECT TO_DATE(HIREDATE,'DD-MM-YY') FROM EMP;
```

> [16:14:04.107]java.sql.SQLDataException: JDBC-5010:Format ends before the entire input string is converted.

→ FORMAT이 맞지 않아 오류가 뜬다.

## 3. 정의된 날짜를 지정(2)

```
SELECT TO_DATE(HIREDATE,'YYYY/MM/DD') FROM EMP;
```

| # | TO_DATE(H... |
|---|---|
| 1 | 1995/12/19 |
| 2 | 1996/03/09 |
| 3 | 1996/11/05 |
| 4 | 1997/04/11 |
| 5 | 1997/05/13 |
| 6 | 1998/05/03 |
| 7 | 1998/07/07 |
| 8 | 1998/12/29 |
| 9 | 1999/06/20 |
| 10 | 1999/10/17 |
| 11 | 2000/05/12 |
| 12 | 2000/12/08 |
| 13 | 2001/03/01 |
| 14 | 2002/07/01 |
| 15 | 2002/09/01 |

## 4. 정의된 날짜를 지정(3)

```
SELECT TO_CHAR(HIREDATE,'DD-MM-YY') FROM EMP;
```

| # | TO_CHAR(H... |
|---|---|
| 1 | 19-12-95 |
| 2 | 09-03-96 |
| 3 | 05-11-96 |
| 4 | 11-04-97 |
| 5 | 13-05-97 |
| 6 | 03-05-98 |
| 7 | 07-07-98 |
| 8 | 29-12-98 |
| 9 | 20-06-99 |
| 10 | 17-10-99 |
| 11 | 12-05-00 |
| 12 | 08-12-00 |
| 13 | 01-03-01 |
| 14 | 01-07-02 |
| 15 | 01-09-02 |

→ 날짜형식을 TO_CHAR( )로 바꾸면 포멧 변경 가능

# 23. NULL 치환 함수-NVL2( ), NULLIF( ), COALESCE( )

## 23.1. 수행

| 수행내역 |
| --- |
| NVL2( ) 함수 사용 |
| NVL( ) vs NVL2( ) |
| NULLIF( ) 함수 사용 |
| COALESCE( ) vs NVL( ) |

## 23.2. 결과

### 1. BONUS값이 NULL값이면 N, NULL값이 아니면 Y

```
SELECT BONUS, NVL2(BONUS,'Y','N') AS BONUS_YN
FROM EMP
WHERE DEPT_CD=1000;
```

| # | BONUS | BONUS_YN |
| --- | --- | --- |
| 1 | 1,500 | Y |
| 2 | 2,000 | Y |
| 3 | <NULL> | N |
| 4 | 100 | Y |
| 5 | 1,100 | Y |

### 2. NVL( ) vs NVL2( )

```
SELECT BONUS, NVL(BONUS,'N'), NVL2(BONUS,'Y','N')
FROM EMP
WHERE DEPT_CD=2000;
```

> [16:50:49.700]java.sql.SQLException: JDBC-5074:Given string does not represent a number in proper format.

→ NVL( )함수쓸 때 숫자형인 변수에 있는 NULL값을 문자형으로 지정하려고 하면 ERROR

```
SELECT BONUS, NVL(BONUS,0), NVL2(BONUS,'Y','N')
FROM EMP
WHERE DEPT_CD=2000;
```

| # | BONUS | NVL(BONUS,0) | NVL2(BONU... |
|---|---|---|---|
| 1 | <NULL> | 0 | N |
| 2 | <NULL> | 0 | N |
| 3 | <NULL> | 0 | N |
| 4 | 400 | 400 | Y |
| 5 | 1,000 | 1,000 | Y |
| 6 | 2,000 | 2,000 | Y |

### 3. BONUS가 1500인 값들을 NULL로 치환

```
SELECT BONUS, NULLIF(BONUS,1500)
FROM EMP
WHERE BONUS=1500;
```

| # | BONUS | NULLIF(BO... |
|---|---|---|
| 1 | 1,500 | <NULL> |
| 2 | 1,500 | <NULL> |
| 3 | 1,500 | <NULL> |

### 4. COALESCE( )함수는 NVL( )와 비슷

```
SELECT BONUS, COALESCE(BONUS,0), NVL(BONUS,0)
FROM EMP
WHERE DEPT_CD=1000;
```

| # | BONUS | COALESCE(... | NVL(BONUS,0) |
|---|---|---|---|
| 1 | 1,500 | 1,500 | 1,500 |
| 2 | 2,000 | 2,000 | 2,000 |
| 3 | <NULL> | 0 | 0 |
| 4 | 100 | 100 | 100 |
| 5 | 1,100 | 1,100 | 1,100 |

# 24. REGR_ 함수 (LINEAR REGRESSION)

**y=a+bx**

$$a = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2}$$
$$b = \bar{y} - \bar{x}a$$

# 24.1. 수행

| 수행내역 |
| --- |
| XY 테이블 생성 |
| 회귀직선의 기울기 계산 |
| 회귀직선의 절편 계산 |
| 회귀선 적합에 사용되는 수 |
| 결정계수 |
| X의 평균 |
| Y의 평균 |

# 23.2. 결과

## 1. XY 테이블 생성

```
CREATE TABLE XY(
X NUMBER(10),
Y NUMBER(10));

INSERT INTO XY VALUES(1,1);
INSERT INTO XY VALUES(2,3);
INSERT INTO XY VALUES('',5);
INSERT INTO XY VALUES(8,2);
INSERT INTO XY VALUES('','');

SELECT * FROM XY;
```

| # | X | Y |
| --- | --- | --- |
| 1 | 1 | 1 |
| 2 | 2 | 3 |
| 3 | <NULL> | 5 |
| 4 | 8 | 2 |
| 5 | <NULL> | <NULL> |

## 2. 회귀직선의 기울기 계산

```
/*REGR_SLOPE(종속변수,독립변수)*/
SELECT REGR_SLOPE(Y,X) FROM XY;
```

| # | REGR_SLOP... |
|---|---|
| 1 | 0.0348837... |

```
/*독립변수, 종속변수 바꿔서 구하기*/
SELECT REGR_SLOPE(X,Y) FROM XY;
```

| # | REGR_SLOP... |
|---|---|
| 1 | 0.5 |

## 3. 회귀직선의 절편 계산

```
SELECT REGR_INTERCEPT(Y,X) FROM XY;
```

| # | REGR_INTE... |
|---|---|
| 1 | 1.8720930... |

## 4. 회귀선 적합에 사용되는 수

```
SELECT REGR_COUNT(Y,X) FROM XY;
```

| # | REGR_COUN... |
|---|---|
| 1 | 3 |

→ NULL값이 포함된 것 제외

## 5. 결정계수 구하기

```
/*REGR_R2(종속변수,독립변수)*/
SELECT REGR_R2(Y,X) FROM XY;
```

| # | REGR_R2(Y,X) |
|---|---|
| 1 | 0.0174418... |

## 6. X의 평균 구하기

```
SELECT REGR_AVGX(Y,X) FROM XY;
```

| # | REGR_AVGX... |
|---|---|
| 1 | 3.6666666... |

## 7. Y의 평균 구하기

```
SELECT REGR_AVGY(Y,X) FROM XY;
```

| # | REGR_AVGY... |
|---|---|
| 1 | 2 |