**2. SQL SELECT 문 작성**

|  |  |  |
| --- | --- | --- |
| 실습 2-1 | | 테이블의 모든 컬럼 검색 |
| 01  02 | SELECT \*  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-2 | | 테이블의 모든 컬럼 검색 |
| 01  02 | SELECT \*  FROM departments ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-3 | | 테이블 종류 확인 |
| 01  02 | SELECT \*  FROM tab ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-4 | | 특정 컬럼 검색 |
| 01  02 | SELECT employee\_id, last\_name, salary, department\_id  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-5 | | 컬럼 구성 확인 |
| 01  02  03 | DESCRIBE employees  또는  DESC employees | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-6 | | 산술식 사용 |
| 01  02 | SELECT employee\_id, last\_name, salary, salary / 4, salary \* 12  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-7 | | 연산자 우선 순위 확인 |
| 01  02 | SELECT employee\_id, last\_name, salary + 1000 / 4, (salary + 1000) / 4  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-8 | | 문자에서 산술 연산 |
| 01  02 | SELECT employee\_id, first\_name + last\_name  FROM employees ; | |
| ORA-01722: 수치가 부적합합니다  01722. 00000 - "invalid number" | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-9 | | 날짜에서 산술 연산 |
| 01  02 | SELECT employee\_id, last\_name, hire\_date, hire\_date \* 2  FROM employees ; | |
| ORA-00932: 일관성 없는 데이터 유형: NUMBER이(가) 필요하지만 DATE임  00932. 00000 - "inconsistent datatypes: expected %s got %s" | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-10 | | 날짜에서 산술 연산 |
| 01  02 | SELECT employee\_id, last\_name, hire\_date, hire\_date + 100,  hire\_date - 100  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-11 | | 날짜 연산 |
| 01  02  03  04 | ALTER SESSION SET nls\_date\_format = 'YYYY/MM/DD HH24:MI:SS' ;  SELECT SYSDATE  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-12 | | 날짜 연산 |
| 01  02  03  04 | ALTER SESSION SET nls\_date\_format = 'YYYY/MM/DD' ;  SELECT SYSDATE  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-13 | | YY 포맷 주의 |
| 01  02  03 | SELECT TO\_DATE('95/11/25','YY/MM/DD')  ,TO\_DATE('95/11/25','RR/MM/DD')  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-14 | | 날짜 연산 |
| 01  02 | SELECT last\_name, hire\_date, SYSDATE - hire\_date  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-15 | | 날짜 연산 |
| 01  02 | SELECT last\_name, hire\_date, SYSDATE + hire\_date  FROM employees ; | |
| ORA-00975: 날짜와 날짜의 가산은 할 수 없습니다  00975. 00000 - "date + date not allowed" | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-16 | | 날짜 연산 |
| 01  02  03  04 | ALTER SESSION SET nls\_date\_format = 'YYYY/MM/DD HH24:MI:SS' ;  SELECT SYSDATE, SYSDATE + 10, SYSDATE - 10  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-17 | | 날짜 연산 |
| 01  02 | SELECT SYSDATE, SYSDATE + 1/24, SYSDATE + 5/1440  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-18 | | 날짜 연산 |
| 01  02  03  04 | SELECT SYSDATE, SYSDATE - 5/24, SYSDATE - 50/1440  FROM dual ;  ALTER SESSION SET nls\_date\_format = 'YYYY/MM/DD' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-19 | | 연결 연산자 |
| 01  02 | SELECT employee\_id, first\_name || last\_name  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-20 | | 연결 연산자 |
| 01  02 | SELECT employee\_id, first\_name || ' ' || last\_name  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-21 | | 리터럴 값 |
| 01  02 | SELECT employee\_id, last\_name, salary \* 12, 'ABC', '2021-01-01'  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-22 | | 리터럴 값 |
| 01  02 | SELECT employee\_id, 'last\_name'  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-23 | | NULL 확인 |
| 01  02 | SELECT cust\_id, lname, gender, address  FROM custs ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-24 | | NULL 확인 (SQL Developer 환경 설정 변경 후) |
| 01  02 | SELECT cust\_id, lname, gender, address  FROM custs ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-25 | | 산술 연산에서의 NULL |
| 01  02  03 | SELECT employee\_id, last\_name, salary, commission\_pct,  (salary\*12) + ((salary\*12)\*commission\_pct)  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-26 | | 컬럼 별칭 필요 |
| 01  02 | SELECT employee\_id, last\_name, salary,  commission\_pct, (salary\*12)\*commission\_pct  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-27 | | 컬럼 별칭 |
| 01  02 | SELECT cust\_id, fname first\_name, lname last\_name, email cust\_email  FROM custs ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-28 | | 컬럼 별칭 |
| 01  02 | SELECT cust\_id, fname AS first\_name, lname AS last\_name, email AS cust\_email  FROM custs ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-29 | | 컬럼 별칭 |
| 01  02  03  04 | SELECT employee\_id AS "EMPID"  ,first\_name||' '||last\_name AS "Name"  ,salary \* 12 AS "Annual Salary"  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-30 | | 중복 행 |
| 01  02 | SELECT department\_id, job\_id  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-31 | | 중복 행 제거 |
| 01  02 | SELECT DISTINCT department\_id, job\_id  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-32 | | 중복 행 제거 |
| 01  02 | SELECT UNIQUE department\_id, job\_id  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-33 | | DISTINCT 주의 사항 |
| 01  02 | SELECT department\_id, DISTINCT job\_id  FROM employees ; | |
| ORA-00936: 누락된 표현식  00936. 00000 - "missing expression" | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-34 | | WHERE 절 사용 |
| 01  02  03 | SELECT cust\_id, lname, birthday,city, country  FROM custs  WHERE cust\_id = 347 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-35 | | 문자 비교 |
| 01  02  03 | SELECT cust\_id, lname, birthday,city, country  FROM custs  WHERE lname = 'Quinlan' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-36 | | 주의 사항 |
| 01  02  03 | SELECT cust\_id, lname, birthday,city, country  FROM custs  WHERE lname = quinlan ; | |
| ORA-00904: "QUINLAN": 부적합한 식별자  00904. 00000 - "%s: invalid identifier" | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-37 | | 날짜 비교 |
| 01  02  03 | SELECT cust\_id, lname, birthday,city, country  FROM custs  WHERE birthday = '1959/05/03' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-38 | | 주의 사항 |
| 01  02  03 | SELECT cust\_id, lname, birthday,city, country  FROM custs  WHERE birthday = 1959/05/03 ; | |
| ORA-00932: 일관성 없는 데이터 유형: DATE이(가) 필요하지만 NUMBER임  00932. 00000 - "inconsistent datatypes: expected %s got %s" | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-39 | | 대소 비교 |
| 01  02  03 | SELECT \*  FROM orders  WHERE order\_total > 30000 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-40 | | 문자에서 대소 비교 |
| 01  02  03 | SELECT cust\_id, lname, birthday, city, country  FROM custs  WHERE lname > 'Quinlan' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-41 | | 날짜에서 대소 비교 |
| 01  02  03 | SELECT cust\_id, lname, birthday, city, country  FROM custs  WHERE birthday > '1983/05/01' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-42 | | 같지 않다 비교 |
| 01  02  03 | SELECT \*  FROM orders  WHERE order\_mode != 'online' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-43 | | 숫자 범위 검색 |
| 01  02  03 | SELECT \*  FROM orders  WHERE order\_total BETWEEN 30000 AND 48552 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-44 | | 날짜 범위 검색 |
| 01  02  03 | SELECT \*  FROM orders  WHERE order\_date BETWEEN '2008/01/01' AND '2008/06/30' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-45 | | 범위 검색 주의 사항 |
| 01  02  03 | SELECT \*  FROM orders  WHERE order\_total BETWEEN 48552 AND 30000 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-46 | | 범위 검색 주의 사항 |
| 01  02  03 | SELECT \*  FROM orders  WHERE order\_total BETWEEN 48552 AND 48552 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-47 | | 범위 검색 |
| 01  02  03  04 | SELECT \*  FROM orders  WHERE order\_total >= 30000  AND order\_total <= 48552 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-48 | | IN 연산자 |
| 01  02  03 | SELECT \*  FROM orders  WHERE order\_status IN (0, 8) ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-49 | | 문자에서 IN 연산자 |
| 01  02  03 | SELECT cust\_id, lname, gender, city, country  FROM custs  WHERE country IN ('USA', 'Spain') ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-50 | | IN 연산자 대체 |
| 01  02  03  04 | SELECT cust\_id, lname, gender, city, country  FROM custs  WHERE country = 'USA'  OR country = 'Spain' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-51 | | LIKE 비교 |
| 01  02  03 | SELECT prod\_id, prod\_name, warranty, list\_price  FROM prods  WHERE prod\_name LIKE 'Monitor%' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-52 | | LIKE 비교 |
| 01  02  03 | SELECT prod\_id, prod\_name, warranty, list\_price  FROM prods  WHERE prod\_name LIKE 'Monitor\_\_\_\_\_\_' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-53 | | LIKE 비교 주의 |
| 01  02  03 | SELECT cust\_id, lname, email, birthday, gender  FROM custs  WHERE email LIKE '%\_%' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-54 | | 저장된 \_ 비교 |
| 01  02  03 | SELECT cust\_id, lname, email, birthday, gender  FROM custs  WHERE email LIKE '%\\_%' ESCAPE '\' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-55 | | 저장된 \_ 비교 |
| 01  02  03 | SELECT cust\_id, lname, email, birthday, gender  FROM custs  WHERE email LIKE '%!\_%' ESCAPE '!' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-56 | | 날짜에서 LIKE 비교 |
| 01  02  03 | SELECT cust\_id, lname, email, birthday, gender  FROM custs  WHERE birthday LIKE '197%' ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-57 | | 숫자에서 LIKE 비교 |
| 01  02  03 | SELECT cust\_id, lname, email, birthday, gender  FROM custs  WHERE cust\_id LIKE '7%' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-58 | | NULL 비교 |
| 01  02  03 | SELECT cust\_id, lname, email, birthday, gender  FROM custs  WHERE gender = NULL ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-59 | | NULL 비교 |
| 01  02  03 | SELECT cust\_id, lname, email, birthday, gender  FROM custs  WHERE gender IS NULL ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-60 | | NULL 비교 |
| 01  02  03 | SELECT cust\_id, lname, email, birthday, gender  FROM custs  WHERE gender IS NOT NULL ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-61 | | WHERE 절의 의미 |
| 01  02  03 | SELECT \*  FROM orders  WHERE 1 = 1 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-62 | | WHERE 절의 의미 |
| 01  02  03 | SELECT \*  FROM orders  WHERE 1 = 0 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-63 | | 논리 연산자 |
| 01  02  03  04 | SELECT \*  FROM orders  WHERE order\_mode = 'direct'  order\_total > 30000 ; | |
| ORA-00933: SQL 명령어가 올바르게 종료되지 않았습니다  00933. 00000 - "SQL command not properly ended" | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-64 | | 논리 연산자 AND |
| 01  02  03  04 | SELECT \*  FROM orders  WHERE order\_mode = 'direct'  AND order\_total > 30000 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-65 | | 논리 연산자 OR |
| 01  02  03  04 | SELECT \*  FROM orders  WHERE order\_mode = 'direct'  OR order\_total > 30000 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-66 | | 논리 연산자 NOT |
| 01  02  03 | SELECT \*  FROM orders  WHERE order\_total NOT BETWEEN 10000 AND 70000 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-67 | | 논리 연산자 NOT |
| 01  02  03 | SELECT cust\_id, lname, gender, city, country  FROM custs  WHERE country NOT IN ('USA', 'Spain','United Kingdom') ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-68 | | 논리 연산자 NOT |
| 01  02  03 | SELECT prod\_id, prod\_name, warranty, list\_price  FROM prods  WHERE prod\_name NOT LIKE 'Monitor%' ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-69 | | 논리 연산자 NOT |
| 01  02  03 | SELECT \*  FROM orders  WHERE NOT order\_mode = 'direct' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-70 | | 논리 연산자 우선 순위 |
| 01  02  03  04  05 | SELECT cust\_id, lname, gender, city, country  FROM custs  WHERE country = 'USA'  OR country = 'Spain'  AND gender = 'F' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-71 | | 논리 연산자 우선 순위 |
| 01  02  03  04  05 | SELECT cust\_id, lname, gender, city, country  FROM custs  WHERE ( country = 'USA'  OR country = 'Spain' )  AND gender = 'F' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-72 | | 논리 연산자 우선 순위 |
| 01  02  03  04 | SELECT cust\_id, lname, gender, city, country  FROM custs  WHERE country IN ('USA', 'Spain')  AND gender = 'F' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-73 | | 오름 차순 정렬 |
| 01  02  03 | SELECT cust\_id, lname, birthday, credit\_limit, country  FROM custs  ORDER BY cust\_id ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-74 | | 내림 차순 정렬 |
| 01  02  03 | SELECT cust\_id, lname, birthday, credit\_limit, country  FROM custs  ORDER BY cust\_id DESC ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-75 | | 다중 컬럼 정렬 |
| 01  02  03 | SELECT cust\_id, lname, birthday, credit\_limit, country  FROM custs  ORDER BY country ASC, credit\_limit DESC ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 2-76 | | 컬럼 별칭 및 순서 사용 |
| 01  02  03  04  05  06  07 | SELECT cust\_id AS ID  ,lname AS NAME  ,birthday AS BIRTH  ,credit\_limit AS CREDIT  ,country AS COUNTRY  FROM custs  ORDER BY 5 ASC, credit DESC ; | |
| ... | | |

**3. 단일 행 함수 사용**

|  |  |  |
| --- | --- | --- |
| 실습 3-1 | | 문자 함수 |
| 01  02 | SELECT lname, UPPER(lname), LOWER(lname), INITCAP(LNAME)  FROM custs ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-2 | | 문자 함수 |
| 01  02  03  04 | SELECT lname, LENGTH(lname),  phone, SUBSTR(phone,2,3)  FROM custs  WHERE phone LIKE '(%'; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-3 | | 문자 함수 |
| 01  02  03  04  05 | SELECT email  ,INSTR(email,'@')  ,REPLACE(email,'\_','.')  FROM custs  WHERE email LIKE '%\\_%' ESCAPE '\'; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-4 | | 문자 함수 |
| 01  02  03  04 | SELECT email  ,REPLACE(SUBSTR(email,1, INSTR(email,'@')-1),'\_','.') AS id  FROM custs  WHERE email LIKE '%\\_%' ESCAPE '\'; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-5 | | 숫자 함수 |
| 01  02  03  04  05 | SELECT 58.567  ,ROUND(58.567, 2)  ,ROUND(58.567, 0)  ,ROUND(58.567, -1)  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-6 | | 숫자 함수 |
| 01  02  03  04  05 | SELECT 58.567  ,TRUNC(58.567, 2)  ,TRUNC(58.567, 0)  ,TRUNC(58.567, -1)  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-7 | | dual 테이블 필요성 |
| 01  02  03  04  05 | SELECT 58.567  ,ROUND(58.567, 2)  ,ROUND(58.567, 0)  ,ROUND(58.567, -1)  FROM orders ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-8 | | 날짜 함수 |
| 01  02  03  04 | SELECT order\_date  ,ADD\_MONTHS(order\_date, 2)  ,MONTHS\_BETWEEN(SYSDATE,order\_date)  FROM orders ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-9 | | 날짜 함수 |
| 01  02  03  04 | SELECT order\_date  ,NEXT\_DAY(order\_date, '금요일')  ,LAST\_DAY(order\_date)  FROM orders ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-10 | | 형 변환 함수 |
| 01  02  03  04  05 | SELECT SYSDATE  ,TO\_CHAR(SYSDATE, 'YYYY')  ,TO\_CHAR(SYSDATE, 'MM/DD')  ,TO\_CHAR(SYSDATE, 'Month DD, YYYY')  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-11 | | 형 변환 함수 |
| 01  02  03  04 | SELECT last\_name, salary  ,TO\_CHAR(salary, '$99,999.00')  ,TO\_CHAR(salary, '$00,000.00')  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-12 | | 형 변환 함수 |
| 01  02  03 | SELECT last\_name, salary  ,TO\_CHAR(salary, 'L99,999.00')  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-13 | | 형 변환 함수 |
| 01  02 | SELECT '$15,000' + '$20,000'  FROM dual ; | |
| ORA-01722: 수치가 부적합합니다  01722. 00000 - "invalid number" | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-14 | | 형 변환 함수 |
| 01  02 | SELECT TO\_NUMBER('$15,000','$99,999') + TO\_NUMBER('$20,000','$99,999')  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-15 | | 형 변환 함수 |
| 01  02 | SELECT TO\_DATE('2021/01/01','YYYY/MM/DD')  FROM dual ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-16 | | 형 변환 함수 |
| 01  02  03  04  05  06  07 | ALTER SESSION SET nls\_date\_format = 'YYYY/MM/DD HH24:MI:SS' ;  SELECT '2021/01/01'  ,TO\_DATE('2021/01/01','YYYY/MM/DD')  FROM dual ;  ALTER SESSION SET nls\_date\_format = 'YYYY/MM/DD' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-17 | | NVL 함수 |
| 01  02  03  04 | SELECT last\_name, salary, commission\_pct  ,(salary\*12) + ((salary\*12)\*commission\_pct)  FROM employees  WHERE department\_id IN (50, 80) ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-18 | | NVL 함수 |
| 01  02  03 | SELECT last\_name, commission\_pct, NVL(commission\_pct,0)  FROM employees  WHERE department\_id IN (50, 80) ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-19 | | NVL 함수 |
| 01  02  03  04 | SELECT last\_name, salary, commission\_pct  ,(salary\*12) + ((salary\*12)\*NVL(commission\_pct,0))  FROM employees  WHERE department\_id IN (50, 80) ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-20 | | 조건 문 필요 |
| 01  02  03 | SELECT last\_name, job\_id, salary, salary \* 1.1  FROM employees  WHERE job\_id = 'AD\_PRES' ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-21 | | CASE 문 |
| 01  02  03  04  05  06 | SELECT last\_name, job\_id, salary,  CASE job\_id WHEN 'AD\_PRES' THEN salary \* 1.1  WHEN 'AD\_VP' THEN salary \* 1.15  WHEN 'IT\_PROG' THEN salary \* 1.2  ELSE salary \* 1.05 END AS new\_salary  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-22 | | CASE 문 |
| 01  02  03  04  05  06 | SELECT last\_name, job\_id, salary,  CASE WHEN job\_id = 'AD\_PRES' THEN salary \* 1.1  WHEN job\_id = 'AD\_VP' THEN salary \* 1.15  WHEN job\_id = 'IT\_PROG' THEN salary \* 1.2  ELSE salary \* 1.05 END AS new\_salary  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-23 | | CASE 문 |
| 01  02  03  04  05  06 | SELECT last\_name, job\_id, salary,  CASE WHEN salary <= 5000 THEN '평균 이하'  WHEN salary <= 10000 THEN '평균'  WHEN salary <= 20000 THEN '평균 이상'  ELSE '최상급' END AS grade  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-24 | | DECODE 함수 |
| 01  02  03  04  05  06 | SELECT last\_name, job\_id, salary,  DECODE(job\_id, 'AD\_PRES' ,salary \* 1.1  , 'AD\_VP' ,salary \* 1.15  ,'IT\_PROG' ,salary \* 1.2  ,salary \* 1.05) AS new\_salary  FROM employees ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 3-25 | | DECODE 함수 |
| 01  02  03  04  05 | SELECT last\_name, job\_id, salary,  DECODE(SIGN(salary-20000),1,'최상급',  DECODE(SIGN(salary-10000),1,'평균이상',  DECODE(SIGN(salary-5000),1,'평균','평균이하'))) AS grade  FROM employees ; | |
| ... | | |

**4. 그룹 함수 사용**

|  |  |  |
| --- | --- | --- |
| 실습 4-1 | | 그룹 함수 종류 |
| 01  02 | SELECT SUM(order\_total), AVG(order\_total),  MAX(order\_total), MIN(order\_total)  FROM orders ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-2 | | 그룹 함수와 데이터 타입 |
| 01  02 | SELECT SUM(order\_date)  FROM orders ; | |
| ORA-00932: 일관성 없는 데이터 유형: NUMBER이(가) 필요하지만 DATE임  00932. 00000 - "inconsistent datatypes: expected %s got %s" | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-3 | | 그룹 함수 종류 |
| 01  02 | SELECT MAX(last\_name), MIN(hire\_date)  FROM employees ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-4 | |  |
| 01  02 | SELECT \*  FROM orders ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-5 | | 중복 제거 |
| 01  02 | SELECT COUNT(order\_status), COUNT(DISTINCT order\_status)  FROM orders ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-6 | | COUNT 함수 |
| 01  02 | SELECT COUNT(\*), COUNT(sales\_rep\_id), COUNT(DISTINCT sales\_rep\_id)  FROM orders ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-7 | | NULL 제외 |
| 01  02 | SELECT SUM(commission\_pct)/COUNT(\*)  ,AVG(commission\_pct)  FROM employees ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-8 | | NULL 포함 |
| 01  02 | SELECT AVG(NVL(commission\_pct,0))  FROM employees ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-9 | | GROUP BY 절 필요성 |
| 01  02  03 | SELECT SUM(salary)  FROM employees  WHERE department\_id = 80 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-10 | | GROUP BY 절 사용 |
| 01  02  03 | SELECT SUM(salary)  FROM employees  GROUP BY department\_id ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-11 | | GROUP BY 절 |
| 01  02  03 | SELECT department\_id, SUM(salary)  FROM employees  GROUP BY department\_id ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-12 | | 주의 사항 |
| 01  02  03 | SELECT department\_id, job\_id, SUM(salary)  FROM employees  GROUP BY department\_id ; | |
| ORA-00979: GROUP BY 표현식이 아닙니다.  00979. 00000 - "not a GROUP BY expression" | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-13 | | 주의 사항 |
| 01  02 | SELECT department\_id, SUM(salary)  FROM employees ; | |
| ORA-00937: 단일 그룹의 그룹 함수가 아닙니다  00937. 00000 - "not a single-group group function" | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-14 | | 다중 컬럼을 이용한 GROUP BY |
| 01  02  03 | SELECT department\_id, job\_id, SUM(salary)  FROM employees  GROUP BY department\_id, job\_id ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-15 | |  |
| 01  02  03  04  05 | SELECT TO\_CHAR(order\_date, 'YYYY') AS order\_year  ,cust\_id  ,SUM(order\_total)  FROM orders  GROUP BY TO\_CHAR(order\_date, 'YYYY'), cust\_id ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-16 | | 중복 값이 없는 GROUP BY |
| 01  02  03 | SELECT employee\_id, last\_name, SUM(salary)  FROM employees  GROUP BY employee\_id, last\_name ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-17 | | WHERE 절의 그룹 함수 |
| 01  02  03  04  05  06 | SELECT TO\_CHAR(order\_date, 'YYYY') AS order\_year  ,cust\_id  ,SUM(order\_total)  FROM orders  WHERE SUM(order\_total) > 20000  GROUP BY TO\_CHAR(order\_date, 'YYYY'), cust\_id ; | |
| ORA-00934: 그룹 함수는 허가되지 않습니다  00934. 00000 - "group function is not allowed here" | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-18 | | HAVING 절 |
| 01  02  03  04  05  06 | SELECT TO\_CHAR(order\_date, 'YYYY') AS order\_year  ,cust\_id  ,SUM(order\_total)  FROM orders  GROUP BY TO\_CHAR(order\_date, 'YYYY'), cust\_id  HAVING SUM(order\_total) > 20000 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-19 | | HAVING 절 주의 사항 |
| 01  02  03  04  05 | SELECT department\_id, SUM(salary)  FROM employees  GROUP BY department\_id  HAVING SUM(salary) > 50000  AND hire\_date < TO\_DATE('2000/01/01','YYYY/MM/DD') ; | |
| ORA-00979: GROUP BY 표현식이 아닙니다.  00979. 00000 - "not a GROUP BY expression" | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-20 | |  |
| 01  02  03  04  05 | SELECT department\_id, SUM(salary)  FROM employees  WHERE hire\_date < TO\_DATE('2000/01/01','YYYY/MM/DD')  GROUP BY department\_id  HAVING SUM(salary) > 50000 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-21 | | 그룹 함수 중첩 |
| 01  02  03 | SELECT MAX(SUM(order\_total))  FROM orders  GROUP BY cust\_id ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 4-22 | | 그룹 함수 중첩 |
| 01  02  03 | SELECT AVG(MAX(SUM(order\_total)))  FROM orders  GROUP BY cust\_id ; | |
| ORA-00935: 그룹 함수의 중첩된 레벨이 너무 깊습니다.  00935. 00000 - "group function is nested too deeply" | | |

**5. 조인문 작성**

|  |  |  |
| --- | --- | --- |
| 실습 5-1 | | 주문번호 2390 정보 확인 => cust\_id |
| 01  02  03 | SELECT \*  FROM orders  WHERE order\_id = 2390 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-2 | | 고객번호 152 정보 확인 |
| 01  02  03 | SELECT cust\_id, lname, city, country  FROM custs  WHERE cust\_id = 152 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-3 | | 주문 상세 확인 |
| 01  02  03 | SELECT \*  FROM order\_items  WHERE order\_id = 2390; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-4 | | 상품 정보 확인 |
| 01  02  03 | SELECT prod\_id, prod\_name, list\_price  FROM prods  WHERE prod\_id IN (1910,1912,1948) ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-5 | | 조인문 확인 |
| 01  02  03  04  05  06  07  08  09  10  11  11  12  13  14  15  16  17  18  19 | SELECT o.order\_id AS order\_id  ,c.cust\_id AS cust\_id  ,c.fname||' '||c.lname AS cust\_name  ,o.order\_date AS order\_date  ,p.prod\_name AS prod\_name  ,p.list\_price AS list\_price  ,i.unit\_price AS price  ,i.quantity AS qty  ,p.list\_price\*i.quantity AS list\_tot  ,i.unit\_price\*i.quantity AS order\_tot  ,p.list\_price\*i.quantity - i.unit\_price\*i.quantity AS discount  ,LPAD(ROUND((1-(i.unit\_price/p.list\_price))\*100,2)||'%',6,' ') AS discount\_rate  FROM custs c  ,orders o  ,order\_items i  ,prods p  WHERE c.cust\_id = o.cust\_id  AND o.order\_id = i.order\_id  AND i.prod\_id = p.prod\_id  AND o.order\_id = 2390 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-6 | | Cartesian Product 생성 |
| 01  02  03 | SELECT \*  FROM orders  ,order\_items ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-7 | | Equi Join |
| 01  02  03  04 | SELECT \*  FROM orders o  ,order\_items i  WHERE o.order\_id = i.order\_id ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-8 | | 컬럼 정의 필요 |
| 01  02  03  04 | SELECT \*  FROM orders  ,order\_items  WHERE order\_id = order\_id ; | |
| ORA-00918: 열의 정의가 애매합니다  00918. 00000 - "column ambiguously defined" | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-9 | | Equi Join |
| 01  02  03  04 | SELECT \*  FROM orders o  ,order\_items i  WHERE o.order\_id = i.order\_id ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-10 | | 일반 조건 추가 |
| 01  02  03  04  05  06 | SELECT o.order\_id, o.order\_date, o.order\_mode, o.cust\_id,  i.prod\_id, i.unit\_price, i.quantity  FROM orders o  ,order\_items i  WHERE o.order\_id = i.order\_id  AND o.order\_id = 2390 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-11 | | 논리 연산자 OR 사용 (일반 조건 추가시) |
| 01  02  03  04  05  06 | SELECT o.order\_id, o.order\_date, o.cust\_id,  i.order\_id, i.prod\_id, i.unit\_price, i.quantity  FROM orders o  ,order\_items i  WHERE o.order\_id = i.order\_id  OR o.order\_id = 2390 ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-12 | | Non-equi join |
| 01  02  03  04  05 | SELECT e.employee\_id, e.last\_name, e.job\_id, e.salary,  j.grade\_level, j.lowest\_sal, j.highest\_sal  FROM employees e  ,job\_grades j  WHERE e.salary BETWEEN j.lowest\_sal AND j.highest\_sal ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-13 | | Non-equi join |
| 01  02  03  04  05  06 | SELECT p.prod\_id, p.prod\_name, p.list\_price, p.min\_price,  i.order\_id, i.unit\_price, i.quantity  FROM prods p  ,order\_items i  WHERE p.prod\_id = i.prod\_id  AND p.min\_price > i.unit\_price ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-14 | | Self join 준비 |
| 01  02  03 | SELECT last\_name, salary, department\_id  FROM employees e  WHERE e.salary > 20000 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-15 | | Self join 준비 |
| 01  02  03 | SELECT last\_name, salary, department\_id  FROM employees e  WHERE employees.salary > 20000 ; | |
| ORA-00904: "EMPLOYEES"."SALARY": 부적합한 식별자  00904. 00000 - "%s: invalid identifier" | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-16 | | Self join 준비 |
| 01  02  03 | SELECT employee\_id, job\_id, salary  FROM employees  WHERE employee\_id = 151 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-17 | | Self join 준비 |
| 01  02  03  04 | SELECT employee\_id, last\_name, hire\_date, job\_id, salary  FROM employees  WHERE job\_id = 'SA\_REP'  AND salary > 9500 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-18 | | Self join |
| 01  02  03  04  05  06 | SELECT e.employee\_id, e.last\_name, e.hire\_date, e.job\_id, e.salary  FROM employees e  ,employees s  WHERE e.job\_id = s.job\_id  AND e.salary > s.salary  AND s.employee\_id = 151 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-19 | | Subquery 활용 |
| 01  02  03  04  05  06  07  08 | SELECT employee\_id, last\_name, hire\_date, job\_id, salary  FROM employees  WHERE job\_id = (SELECT job\_id  FROM employees  WHERE employee\_id = 151)  AND salary > (SELECT salary  FROM employees  WHERE employee\_id = 151) ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-20 | | Join 필요성 |
| 01  02  03  04  05  06 | SELECT e.employee\_id, e.last\_name, e.hire\_date, e.job\_id, e.salary, s.salary  FROM employees e  ,employees s  WHERE e.job\_id = s.job\_id  AND e.salary > s.salary  AND s.employee\_id = 151 ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-21 | | Outer join 준비 |
| 01  02  03  04 | SELECT c.cust\_id, c.lname, o.order\_id, o.order\_date, o.order\_total  FROM custs c  ,orders o  WHERE c.cust\_id = o.cust\_id ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-22 | | Outer join |
| 01  02  03  04 | SELECT c.cust\_id, c.lname, c.country, o.order\_id, o.order\_date, o.order\_total  FROM custs c  ,orders o  WHERE c.cust\_id = o.cust\_id (+); | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-23 | | Outer join |
| 01  02  03  04  05 | SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id,  d.department\_id, d.department\_name  FROM employees e  ,departments d  WHERE e.department\_id = d.department\_id (+) ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-24 | | Outer join |
| 01  02  03  04  05 | SELECT d.department\_id, d.department\_name,  e.employee\_id, e.last\_name, e.salary, e.department\_id  FROM employees e  ,departments d  WHERE e.department\_id (+) = d.department\_id; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-25 | | Outer join 주의 사항 |
| 01  02  03  04  05 | SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id,  d.department\_id, d.department\_name  FROM employees e  ,departments d  WHERE e.department\_id (+) = d.department\_id (+); | |
| ORA-01468: outer-join된 테이블은 1개만 지정할 수 있습니다  01468. 00000 - "a predicate may reference only one outer-joined table" | | |

|  |  |  |
| --- | --- | --- |
| 실습 5-26 | | ANSI Join (FULL OUTER JOIN) |
| 01  02  03  04 | SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id,  d.department\_id, d.department\_name  FROM employees e FULL OUTER JOIN departments d  ON e.department\_id = d.department\_id ; | |
| ... | | |

**ORACLE JOIN vs ANSI JOIN**

|  |
| --- |
| SELECT o.order\_id, o.order\_date, o.order\_mode, o.cust\_id,  i.prod\_id, i.unit\_price, i.quantity  FROM orders o  ,order\_items i  WHERE o.order\_id = i.order\_id ;  SELECT o.order\_id, o.order\_date, o.order\_mode, o.cust\_id,  i.prod\_id, i.unit\_price, i.quantity  FROM orders o  JOIN order\_items i  ON o.order\_id = i.order\_id ; |
| ---------------------------------------------------------------------------------------- |
| SELECT o.order\_id, o.order\_date, o.order\_mode, o.cust\_id,  i.prod\_id, i.unit\_price, i.quantity  FROM orders o  ,order\_items i  WHERE o.order\_id = i.order\_id  AND o.order\_id = 2390 ;  SELECT o.order\_id, o.order\_date, o.order\_mode, o.cust\_id,  i.prod\_id, i.unit\_price, i.quantity  FROM orders o  JOIN order\_items i  ON o.order\_id = i.order\_id  WHERE o.order\_id = 2390 ;  SELECT o.order\_id, o.order\_date, o.order\_mode, o.cust\_id,  i.prod\_id, i.unit\_price, i.quantity  FROM orders o  JOIN order\_items i  ON o.order\_id = i.order\_id  AND o.order\_id = 2390 ; |
| ---------------------------------------------------------------------------------------- |
| SELECT e.employee\_id, e.last\_name, e.job\_id, e.salary,  j.grade\_level, j.lowest\_sal, j.highest\_sal  FROM employees e  ,job\_grades j  WHERE e.salary BETWEEN j.lowest\_sal AND j.highest\_sal ;  SELECT e.employee\_id, e.last\_name, e.job\_id, e.salary,  j.grade\_level, j.lowest\_sal, j.highest\_sal  FROM employees e  JOIN job\_grades j  ON e.salary BETWEEN j.lowest\_sal AND j.highest\_sal ; |
|  |
| SELECT p.prod\_id, p.prod\_name, p.list\_price, p.min\_price,  i.order\_id, i.unit\_price, i.quantity  FROM prods p  ,order\_items i  WHERE p.prod\_id = i.prod\_id  AND p.min\_price > i.unit\_price ;  SELECT p.prod\_id, p.prod\_name, p.list\_price, p.min\_price,  i.order\_id, i.unit\_price, i.quantity  FROM prods p  JOIN order\_items i  ON p.prod\_id = i.prod\_id  AND p.min\_price > i.unit\_price ; |
| ---------------------------------------------------------------------------------------- |
| SELECT o.order\_id AS order\_id  ,c.cust\_id AS cust\_id  ,c.fname||' '||c.lname AS cust\_name  ,o.order\_date AS order\_date  ,o.order\_total AS order\_total  ,i.prod\_id AS prod\_id  ,i.unit\_price AS unit\_price  ,i.quantity AS quantity  FROM custs c  JOIN orders o  JOIN order\_items i  ON c.cust\_id = o.cust\_id  AND o.order\_id = i.order\_id  AND o.order\_id = 2390 ;  오류 발생 명령행: 14 열: 24  오류 보고 -  SQL 오류: ORA-00905: 누락된 키워드  00905. 00000 - "missing keyword" |
| ---------------------------------------------------------------------------------------- |
| SELECT o.order\_id AS order\_id  ,c.cust\_id AS cust\_id  ,c.fname||' '||c.lname AS cust\_name  ,o.order\_date AS order\_date  ,o.order\_total AS order\_total  ,i.prod\_id AS prod\_id  ,i.unit\_price AS unit\_price  ,i.quantity AS quantity  FROM custs c  JOIN orders o  ON c.cust\_id = o.cust\_id  JOIN order\_items i  ON o.order\_id = i.order\_id  WHERE o.order\_id = 2390 ; |
|  |
| SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id,  d.department\_id, d.department\_name  FROM employees e  ,departments d  WHERE e.department\_id = d.department\_id (+) ;  SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id,  d.department\_id, d.department\_name  FROM employees e LEFT OUTER JOIN departments d  ON e.department\_id = d.department\_id ; |
| ---------------------------------------------------------------------------------------- |
| SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id,  d.department\_id, d.department\_name  FROM employees e RIGHT OUTER JOIN departments d  ON e.department\_id = d.department\_id ;  SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id,  d.department\_id, d.department\_name  FROM employees e  ,departments d  WHERE e.department\_id (+) = d.department\_id ; |
| ---------------------------------------------------------------------------------------- |
| SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id,  d.department\_id, d.department\_name  FROM employees e FULL OUTER JOIN departments d  ON e.department\_id = d.department\_id ; |
| ---------------------------------------------------------------------------------------- |
| SELECT \*  FROM orders  ,order\_items ;  SELECT \*  FROM orders CROSS JOIN order\_items ; |

**6. Subquery**

|  |  |  |
| --- | --- | --- |
| 실습 6-1 | | Subquery 준비 |
| 01  02  03 | SELECT salary  FROM employees  WHERE employee\_id = 151 ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-2 | | Subquery 준비 |
| 01  02  03 | SELECT employee\_id, last\_name, hire\_date, job\_id, salary  FROM employees  WHERE salary > 9500 ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-3 | | Subquery 활용 |
| 01  02  03  04  05 | SELECT employee\_id, last\_name, hire\_date, job\_id, salary  FROM employees  WHERE salary > (SELECT salary  FROM employees  WHERE employee\_id = 151) ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-4 | | Subquery 활용 |
| 01  02  03  04  05  06  07  08 | SELECT employee\_id, last\_name, hire\_date, job\_id, salary  FROM employees  WHERE job\_id = (SELECT job\_id  FROM employees  WHERE employee\_id = 151)  AND salary > (SELECT salary  FROM employees  WHERE employee\_id = 151) ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-5 | | Subquery 활용 |
| 01  02  03  04  05 | SELECT prod\_id, SUM(quantity)  FROM order\_items  GROUP BY prod\_id  HAVING SUM(quantity) = (SELECT MAX(SUM(quantity))  FROM order\_items  GROUP BY prod\_id) ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-6 | | Subquery 활용 |
| 01  02  03  04 | SELECT cust\_id, lname, gender, country  FROM custs  WHERE cust\_id = (SELECT cust\_id  FROM orders) ; | |
| ORA-01427: 단일 행 하위 질의에 2개 이상의 행이 리턴되었습니다.  01427. 00000 - "single-row subquery returns more than one row" | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-7 | | 다중 행 Subquery |
| 01  02  03  04 | SELECT cust\_id, lname, gender, country  FROM custs  WHERE cust\_id = (SELECT cust\_id  FROM orders) ; | |
| ORA-01427: 단일 행 하위 질의에 2개 이상의 행이 리턴되었습니다.  01427. 00000 - "single-row subquery returns more than one row" | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-8 | | 다중 행 Subquery |
| 01  02  03  04 | SELECT cust\_id, lname, gender, country  FROM custs  WHERE cust\_id IN (SELECT cust\_id  FROM orders) ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-9 | | 다중 행 Subquery |
| 01  02  03  04  05 | SELECT employee\_id, last\_name, salary, department\_id  FROM employees  WHERE salary > (SELECT AVG(salary)  FROM employees  GROUP BY department\_id) ; | |
| ORA-01427: 단일 행 하위 질의에 2개 이상의 행이 리턴되었습니다.  01427. 00000 - "single-row subquery returns more than one row" | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-10 | | 다중 행 Subquery |
| 01  02  03  04  05 | SELECT employee\_id, last\_name, salary, department\_id  FROM employees  WHERE salary > ANY (SELECT AVG(salary)  FROM employees  GROUP BY department\_id) ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-11 | | 다중 행 Subquery |
| 01  02  03  04  05 | SELECT employee\_id, last\_name, salary, department\_id  FROM employees  WHERE salary > (SELECT MIN(AVG(salary))  FROM employees  GROUP BY department\_id) ; | |

|  |  |  |
| --- | --- | --- |
| 실습 6-12 | | 다중 행 Subquery |
| 01  02  03  04  05 | SELECT employee\_id, last\_name, salary, department\_id  FROM employees  WHERE salary > ALL (SELECT AVG(salary)  FROM employees  GROUP BY department\_id) ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-13 | | 다중 행 Subquery |
| 01  02  03  04  05 | SELECT employee\_id, last\_name, salary, department\_id  FROM employees  WHERE salary > (SELECT MAX(AVG(salary))  FROM employees  GROUP BY department\_id) ; | |

|  |  |  |
| --- | --- | --- |
| 실습 6-14 | | 다중 컬럼 Subquery |
| 01  02  03  04  05 | SELECT employee\_id, last\_name, salary, department\_id  FROM employees  WHERE (department\_id, salary) IN (SELECT department\_id, MIN(salary)  FROM employees  GROUP BY department\_id) ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-15 | | Correlated Subquery |
| 01  02  03  04  05 | SELECT employee\_id, last\_name, salary, department\_id  FROM employees e  WHERE salary > (SELECT AVG(salary)  FROM employees  WHERE department\_id = e.department\_id) ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 6-16 | | 다중 컬럼 Subquery |
| 01  02  03  04  05  06  07 | SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id, ROUND(a.avgsal)  FROM employees e  ,(SELECT department\_id, AVG(salary) AS AVGSAL  FROM employees  GROUP BY department\_id) a  WHERE e.department\_id = a.department\_id  AND e.salary > a.avgsal ; | |
| ... | | |

**7. 집합 연산자**

|  |  |  |
| --- | --- | --- |
| 실습 7-1 | | UNION 연산자 |
| 01  02  03  04  05 | SELECT job\_id  FROM employees  UNION  SELECT job\_id  FROM retired\_employees; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 7-2 | | UNION ALL 연산자 |
| 01  02  03  04  05  06 | SELECT job\_id, department\_id  FROM employees  UNION ALL  SELECT job\_id, department\_id  FROM retired\_employees  ORDER BY job\_id; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 7-3 | | INTERSECT 연산자 |
| 01  02  03  04  05 | SELECT employee\_id  FROM employees  INTERSECT  SELECT manager\_id  FROM departments; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 7-4 | | MINUS 연산자 |
| 01  02  03  04  05 | SELECT empno  FROM emp  MINUS  SELECT mgr  FROM emp; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 7-5 | | MINUS 연산자 |
| 01  02  03  04  05 | SELECT department\_id  FROM departments  MINUS  SELECT department\_id  FROM employees; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 7-6 | | 사원의 회사 직무 이동 이력 확인 |
| 01  02  03  04  05  06  07  08 | SELECT employee\_id, NULL start\_date, NULL end\_date, job\_id, department\_id  FROM employees  WHERE employee\_id = 176  UNION ALL  SELECT employee\_id, start\_date, end\_date, job\_id, department\_id  FROM job\_history  WHERE employee\_id = 176  ORDER BY start\_date; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 7-7 | | 집합 연산자 활용 |
| 01  02  03  04  05  06  07  08 | SELECT department\_id, job\_id, SUM(salary) AS SUMSAL  FROM employees  GROUP BY department\_id, job\_id  UNION ALL  SELECT department\_id, NULL, SUM(salary) AS SUMSAL  FROM employees  GROUP BY department\_id  ORDER BY department\_id, job\_id ; | |
| ... | | |

**8. DML 명령문**

|  |  |  |
| --- | --- | --- |
| 실습 8-1 | | INSERT 문 |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | INSERT INTO emp  VALUES (9999,'KIM','SALESMAN',7839,TO\_DATE('2021/03/20','YYYY/MM/DD'),2500,1000,40);  INSERT INTO emp (empno, ename, sal)  VALUES (9998,'LEE',2000);  INSERT INTO emp  VALUES (9997,'KANG',NULL,NULL,NULL,NULL,NULL,NULL);  INSERT INTO emp (empno, ename, job, hiredate, sal, comm, deptno)  SELECT employee\_id, last\_name, job\_id, hire\_date, salary, NULL , 10  FROM employees  WHERE job\_id = 'IT\_PROG';  SELECT \*  FROM emp ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 8-2 | | UPDATE 문 |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | UPDATE emp  SET sal = sal \* 1.1  WHERE deptno = 10 ;  UPDATE emp  SET hiredate = SYSDATE ;  UPDATE emp  SET (job,sal) = (SELECT job\_id,salary  FROM employees  WHERE employee\_id = 205)  WHERE empno = 9999;  SELECT \*  FROM emp ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 8-3 | | DELETE 문 |
| 01  02  03  04  05 | DELETE emp  WHERE deptno IS NULL ;  SELECT \*  FROM emp ; | |
| ... | | |

|  |  |  |
| --- | --- | --- |
| 실습 8-4 | | DELETE 문 – 조건에 해당하는 행 삭제 |
| 01  02  03  04 | DELETE emp ;  SELECT \*  FROM emp ; | |
|  | | |

|  |  |  |
| --- | --- | --- |
| 실습 8-5 | | 트랜잭션 제어 |
| 01  02  03  04 | ROLLBACK ;  SELECT \*  FROM emp ; | |
|  | | |
| 01  02  03  04  05  06  07  08  09 | INSERT INTO emp (empno, ename, sal, deptno) VALUES (1111, 'KIM', 2000, 20) ;  COMMIT ;  ROLLBACK ;  SELECT \*  FROM emp ; | |
| ... | | |
| 01  02  03  04  05  06  07 | DELETE emp  WHERE empno = 1111 ;  COMMIT ;  SELECT \*  FROM emp ; | |
|  | | |

**9. DDL 명령문**

|  |  |  |
| --- | --- | --- |
| 실습 9-1 | | 테이블 생성 및 확인 |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | CREATE TABLE copy\_emp  (empno NUMBER(4),  name VARCHAR2(10),  hiredate DATE,  sal NUMBER(7,2),  DEPTNO NUMBER(2),  mobile NUMBER(11));  DESC copy\_emp ;  SELECT \*  FROM copy\_emp ;  DROP TABLE copy\_emp ; | |

|  |  |  |
| --- | --- | --- |
| 실습 9-2 | | SUBQUERY를 사용하여 테이블 생성 |
| 01  02  03  04  05  06  07 | CREATE TABLE copy\_emp  AS  SELECT \*  FROM emp;  SELECT \*  FROM copy\_emp ; | |

|  |  |  |
| --- | --- | --- |
| 실습 9-3 | | 테이블 수정 |
| 01  02  03  04  05  06 | ALTER TABLE copy\_emp  ADD (mobile NUMBER(11));  ALTER TABLE copy\_emp READ ONLY;  ALTER TABLE copy\_emp READ WRITE; | |

|  |  |  |
| --- | --- | --- |
| 실습 9-4 | | 테이블 삭제 |
| 01  02  03  04  05  06  07  08  09  10 | DROP TABLE copy\_emp ;  FLASHBACK TABLE copy\_emp TO BEFORE DROP ;  SELECT \*  FROM copy\_emp ;  DROP TABLE copy\_emp PURGE;  FLASHBACK TABLE copy\_emp TO BEFORE DROP ; | |

**10. 제약 조건 소개**

|  |  |  |
| --- | --- | --- |
| 실습 10-1 | | 제약조건 테스트 |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | CREATE TABLE t\_dept  (deptno NUMBER(2) CONSTRAINT t\_dept\_deptno\_pk PRIMARY KEY,  dname VARCHAR2(10) CONSTRAINT t\_dept\_dname\_nn NOT NULL,  loc VARCHAR2(10) DEFAULT 'SEOUL');  INSERT INTO t\_dept VALUES(10, 'SALES', 'SEOUL');  SELECT \*  FROM t\_dept;  -- PRIMARY KEY 제약조건 확인  INSERT INTO t\_dept VALUES(10, 'ACCOUNT', 'SEOUL');  INSERT INTO t\_dept(dname, loc) VALUES('ACCOUNT', 'SEOUL');  -- NOT NULL 제약조건 확인  INSERT INTO t\_dept(deptno, loc) VALUES(20, 'BOSTON');  CREATE TABLE t\_emp  (empno NUMBER(4) CONSTRAINT t\_emp\_empno\_pk PRIMARY KEY,  ename VARCHAR2(10) CONSTRAINT t\_emp\_ename\_nn NOT NULL,  sal NUMBER(7,2) CONSTRAINT t\_emp\_sal\_ck CHECK(sal > 0),  hp VARCHAR(13) CONSTRAINT t\_emp\_hp\_uk UNIQUE,  deptno NUMBER(2) CONSTRAINT t\_emp\_deptno\_fk REFERENCES t\_dept(deptno));  -- UNIQUE 제약조건 확인  INSERT INTO t\_emp VALUES(1, 'KIM', 500, '010-1234-5678',10);  INSERT INTO t\_emp VALUES(2, 'LEE', 450, '010-1234-5678',10);  INSERT INTO t\_emp VALUES(2, 'LEE', 450, NULL,10);  -- check 제약조건 확인  INSERT INTO t\_emp VALUES(3, 'JUNG', 0, '010-1234-8765',10);  -- Foreign key 제약조건 확인  INSERT INTO t\_emp VALUES(3, 'JUNG', 0, '010-1234-8765',20);  DELETE t\_dept WHERE deptno = 10 ;  DROP TABLE t\_emp PURGE;  DROP TABLE t\_dept PURGE; | |

**11. 기타 객체 관리**

|  |  |  |
| --- | --- | --- |
| 실습 11-1 | | SEQUENCE 테스트 |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | CREATE SEQUENCE dept\_sq  START WITH 50  INCREMENT BY 10  MAXVALUE 90 ;  INSERT INTO dept VALUES (dept\_sq.NEXTVAL, 'MARKETING', 'SEOUL');  SELECT \*  FROM dept;  SELECT dept\_sq.NEXTVAL  FROM dual;  SELECT dept\_sq.NEXTVAL  FROM dual;  SELECT dept\_sq.CURRVAL  FROM dual;  SELECT dept\_sq.CURRVAL  FROM dual;  ROLLBACK ;  DROP SEQUENCE dept\_sq ; | |

|  |  |  |
| --- | --- | --- |
| 실습 11-2 | | SYNONYM 테스트 |
| 01  02  03  04  05 | CREATE SYNONYM bigemp for employees ;  SELECT \* FROM bigemp ;  DROP SYNONYM bigemp ; | |

|  |  |  |
| --- | --- | --- |
| 실습 11-3 | | INDEX 테스트 |
| 01  02  03  04  05  06  07  08  09 | CREATE INDEX emp\_job\_idx ON emp(job);  SELECT job, ROWID  FROM emp  ORDER BY 1, 2;    SELECT \*  FROM emp e  WHERE job = 'MANAGER'; | |
|  | | |
| 실습 11-4 | | VIEW 생성 및 조회 |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | CREATE OR REPLACE VIEW v\_test1  AS  SELECT empno, ename, JOB, deptno  FROM emp;  SELECT \*  FROM v\_test1;  DESC v\_test1;  CREATE OR REPLACE VIEW v\_test2  AS  SELECT empno, ename, job, sal\*1.2 cal\_sal, deptno  FROM emp;  SELECT \*  FROM v\_test2; | |

**분석 함수 활용**

SQL은 Relationship 이 존재하는 테이블 구조의 데이터를 제어하기 위한 언어이다. 이러한 SQL은 DB에 저장된 데이터를 제어하기 위한 매우 강력한 언어이지만, 다양한 Business Intelligence Calculation을 수행하기에는 부족함이 존재한다. 때문에 복잡한 형태의 분석 작업을 진행하려면 과도한 프로그래밍이 사용되고, 그로 인해 성능은 저하되기도 한다. Oracle Database 8i부터는 이러한 요구 사항들을 해결하기 위해 새로운 함수를 제공한다. 이 함수들은 분석 작업에 유용하기 때문에 Analytic Functions이라고 하며 DB 버전이 올라갈수록 계속 추가되고 있다.

**SQL> SELECT empno, ename, sal, deptno,**

**SUM(sal) OVER(PARTITION BY deptno) AS dept\_tot**

**FROM emp;**

EMPNO ENAME SAL DEPTNO DEPT\_TOT

---------- ---------- ---------- ---------- ----------

7934 MILLER 1300 10 8750

7782 CLARK 2450 10 8750

7839 KING 5000 10 8750

7902 FORD 3000 20 10875

7788 SCOTT 3000 20 10875

7876 ADAMS 1100 20 10875

7566 JONES 2975 20 10875

7369 SMITH 800 20 10875

7900 JAMES 950 30 9400

7844 TURNER 1500 30 9400

7698 BLAKE 2850 30 9400

7521 WARD 1250 30 9400

7499 ALLEN 1600 30 9400

7654 MARTIN 1250 30 9400

14 rows selected.

분석 함수는 Aggregate Function 뒤에 Analytic Clause(OVER 절)을 통해 행 그룹의 정의를 지정하고 각 그룹당 결과 값을 반복하여 출력한다. 여기서 행 그룹의 범위를 WINDOW라 부르며 하나의 WINDOW가 계산을 수행하는데 사용되는 행들의 집합을 결정하게 되며 PARTITION BY, ORDER BY, WINDOWING을 통하여 조절하게 된다.

분석 함수는 Join 문장, WHERE, GROUP BY, HAVING 등과 함께 쓰일 때 가장 마지막에 연산(집계)을 진행하며 SELECT 절과 ORDER BY 절에서만 사용이 가능하다.

PARTITION BY 절은 GROUP BY 절과 동일한 작업 수행한다. 단, GROUP BY 절을 사용하지 않고 필요한 집합으로 (WINDOW) 행들을 그룹화 시킬 수 있다.

**SQL> SELECT empno, ename, sal, AVG(sal) OVER() AS avg\_overall,**

**AVG(sal) OVER(PARTITION BY deptno) AS avg\_deptno**

**FROM emp ;**

EMPNO ENAME SAL AVG\_OVERALL AVG\_DEPTNO

---------- ---------- ---------- ----------- ----------

7934 MILLER 1300 2073.21429 2916.66667

7782 CLARK 2450 2073.21429 2916.66667

7839 KING 5000 2073.21429 2916.66667

7902 FORD 3000 2073.21429 2175

7788 SCOTT 3000 2073.21429 2175

7876 ADAMS 1100 2073.21429 2175

7566 JONES 2975 2073.21429 2175

7369 SMITH 800 2073.21429 2175

7900 JAMES 950 2073.21429 1566.66667

7844 TURNER 1500 2073.21429 1566.66667

7698 BLAKE 2850 2073.21429 1566.66667

7521 WARD 1250 2073.21429 1566.66667

7499 ALLEN 1600 2073.21429 1566.66667

7654 MARTIN 1250 2073.21429 1566.66667

14 rows selected.

ORDER BY 절은 PARTITION BY로 정의된 WINDOW 내에서의 행들의 정렬 순서를 정의한다.

**SQL> SELECT empno, ename, sal, deptno,**

**row\_number( ) over ( ORDER BY sal ASC ) AS rnum**

**FROM emp ;**

EMPNO ENAME SAL DEPTNO RNUM

---------- ---------- ---------- ---------- ----------

7369 SMITH 800 20 1

7900 JAMES 950 30 2

7876 ADAMS 1100 20 3

7654 MARTIN 1250 30 4

7521 WARD 1250 30 5

7934 MILLER 1300 10 6

7844 TURNER 1500 30 7

7499 ALLEN 1600 30 8

7782 CLARK 2450 10 9

7698 BLAKE 2850 30 10

7566 JONES 2975 20 11

7788 SCOTT 3000 20 12

7902 FORD 3000 20 13

7839 KING 5000 10 14

14 rows selected.

**SQL> SELECT empno, ename, sal, deptno,**

**row\_number( ) over ( PARTITION BY deptno ORDER BY sal DESC ) AS rnum**

**FROM emp ;**

EMPNO ENAME SAL DEPTNO RNUM

---------- ---------- ---------- ---------- ----------

7839 KING 5000 10 1

7782 CLARK 2450 10 2

7934 MILLER 1300 10 3

7902 FORD 3000 20 1

7788 SCOTT 3000 20 2

7566 JONES 2975 20 3

7876 ADAMS 1100 20 4

7369 SMITH 800 20 5

7698 BLAKE 2850 30 1

7499 ALLEN 1600 30 2

7844 TURNER 1500 30 3

7521 WARD 1250 30 4

7654 MARTIN 1250 30 5

7900 JAMES 950 30 6

14 rows selected.

**SQL> SELECT empno, ename, sal, comm,**

**DENSE\_RANK( ) over ( ORDER BY comm ASC ) AS no1,**

**DENSE\_RANK( ) over ( ORDER BY comm ASC NULLS FIRST ) AS no2**

**FROM emp**

**WHERE deptno = 30 ;**

EMPNO ENAME SAL COMM NO1 NO2

---------- ---------- ---------- ---------- ---------- ----------

7698 BLAKE 2850 5 1

7900 JAMES 950 5 1

7844 TURNER 1500 0 1 2

7499 ALLEN 1600 300 2 3

7521 WARD 1250 500 3 4

7654 MARTIN 1250 1400 4 5

6 rows selected.

ORDER BY 절은 PARTITION BY에 의해 그룹화된 행들의 정렬 순서를 결정하며 NULL 값을 가지고 있는 행이 있을 경우 NULL에 대한 값을 FIRST, LAST로 보낼 수 있도록 조절 가능하다.

WINDOWING 절은 일부 Aggregate Function과 함께 쓰일 수 있으며 행들의 그룹을 물리적, 논리적으로 조절하여 Function이 적용될 WINDOW를 정의한다. 즉, PARTITON BY 절은 컬럼에 같은 값을 기준으로만 그룹화하지만 WINDOWING 절은 ROWS와 RANGE를 이용하여 하나의 WINDOW를 결정하는 범위를 보다 자유롭게 조정할 수 있다.

**SQL> SELECT empno, ename, sal,**

**SUM(sal) over ( ORDER BY empno**

**ROWS BETWEEN 1 PRECEDING**

**AND 1 FOLLOWING ) AS physical,**

**SUM(sal) over ( ORDER BY empno**

**RANGE BETWEEN 100 PRECEDING**

**AND 100 FOLLOWING ) AS logical**

**FROM emp**

**WHERE deptno IN (10,20)**

**ORDER BY empno ;**

EMPNO ENAME SAL PHYSICAL LOGICAL

---------- ---------- ---------- ---------- ----------

7369 SMITH 800 3775 800

7566 JONES 2975 6225 2975

7782 CLARK 2450 8425 11550 : EMPNO BETWEEN 7682 AND 7882

7788 SCOTT 3000 10450 11550

7839 KING 5000 9100 15850

7876 ADAMS 1100 9100 15850

7902 FORD 3000 5400 10400

7934 MILLER 1300 4300 10400

8 rows selected.

ROWS는 WINDOW의 범위를 정의할 때 물리적인 행을 지정한다. 어떤 행에서 시작해서 어떤 행 까지가 하나의 WINDOW 영역으로 정의 할지 범위를 BETWEEN을 통하여 정의 할 수 있다. 그리고 추가적으로 UNBOUNDED PRECEDING는 첫 번째 행, UNBOUNDED FOLLOWING은 마지막 행, CURRENT ROW는 현재 행 참조하게 할 수 있다.

RANGE는 논리적인 값을 근거로 WINDOW 범위를 설정할 수 있다. 모든 함수가 WINDOWING절을 사용할 수 있는 것은 아니다. ([매뉴얼](http://docs.oracle.com/cd/E11882_01/server.112/e41084/functions004.htm#SQLRF06174) 참조)

1. **EMP 테이블에서 사원 정보와 소속 부서의 평균 급여를 다음과 같이 검색하시오.**

**검색 결과**

EMPNO ENAME SAL DEPTNO AVG\_SAL

---------- ---------- ---------- ---------- ----------

7782 CLARK 2450 10 2916.67

7934 MILLER 1300 10 2916.67

7839 KING 5000 10 2916.67

7788 SCOTT 3000 20 2175

7566 JONES 2975 20 2175

7369 SMITH 800 20 2175

7876 ADAMS 1100 20 2175

7902 FORD 3000 20 2175

7499 ALLEN 1600 30 1566.67

7521 WARD 1250 30 1566.67

7654 MARTIN 1250 30 1566.67

7698 BLAKE 2850 30 1566.67

7844 TURNER 1500 30 1566.67

7900 JAMES 950 30 1566.67

14 rows selected.

**답안 1. Inline View 활용**

**SQL> SELECT e.empno, e.ename, e.sal, e.deptno, a.avg\_sal**

**FROM emp e, ( SELECT deptno, ROUND(AVG(sal),2) AS AVG\_SAL**

**FROM emp**

**GROUP BY deptno ) a**

**WHERE e.deptno = a.deptno**

**ORDER BY e.deptno ;**

**답안 2. Correlated Subquery 이용**

**SQL> SELECT empno, ename, sal, deptno, (SELECT ROUND(AVG(sal),2)**

**FROM emp**

**WHERE deptno = e.deptno) AS AVG\_SAL**

**FROM emp e**

**ORDER BY deptno ;**

두 가지의 Subquery의 사용 방법을 통해 문제를 해결할 수 있다. 경우에 따라서는 위와 같은 문장의 사용이 필요할 수도 있으나 동일 테이블의 반복적인 접근이 오히려 성능을 저하시키는 경우도 있다.

**답안 3. 분석 함수 사용**

**SQL> SELECT empno, ename, sal, deptno,**

**ROUND(AVG(sal) OVER(PARTITION BY deptno),2) AS AVG\_SAL**

**FROM emp**

**ORDER BY deptno ;**

분석 함수의 사용이 항상 정답은 아니다. 경우에 따라 일반적인 Subquery를 이용하는 것이 필요할 수도 있다. 단, 다양한 문장을 통해 동일한 결과를 검색하도록 문장을 작성할 수 있다면 경우에 따라 필요한 문장의 선택을 보다 손쉽게 할 수 있다. 성능과 관련된 자세한 사항은 SQL Tuning 과정을 참고한다.

1. **EMP 테이블에서 1981년에 입사한 사원 정보와 소속 부서의 평균 급여를 분석 함수를 이용하여 검색하시오.**

**검색 결과**

EMPNO ENAME SAL HIREDATE DEPTNO AVG\_SAL

---------- ---------- ---------- -------- ---------- ----------

7782 CLARK 2450 81/06/09 10 2916.67

7839 KING 5000 81/11/17 10 2916.67

7566 JONES 2975 81/04/02 20 2175

7902 FORD 3000 81/12/03 20 2175

7499 ALLEN 1600 81/02/20 30 1566.67

7521 WARD 1250 81/02/22 30 1566.67

7654 MARTIN 1250 81/09/28 30 1566.67

7698 BLAKE 2850 81/05/01 30 1566.67

7844 TURNER 1500 81/09/08 30 1566.67

7900 JAMES 950 81/12/03 30 1566.67

10 rows selected.

**답안 1. Correlated Subquery 이용**

**SQL> SELECT empno, ename, sal, hiredate, deptno, (SELECT ROUND(AVG(sal),2)**

**FROM emp**

**WHERE deptno = e.deptno) AS AVG\_SAL**

**FROM emp e**

**WHERE hiredate BETWEEN TO\_DATE('1981/01/01','YYYY/MM/DD')**

**AND TO\_DATE('1981/12/31','YYYY/MM/DD')**

**ORDER BY deptno ;**

**답안 2. Inline View 이용**

**SQL> SELECT e.empno, e.ename, e.sal, e.hiredate, e.deptno, a.avg\_sal**

**FROM emp e, ( SELECT deptno, ROUND(AVG(sal),2) AS AVG\_SAL**

**FROM emp**

**GROUP BY deptno ) a**

**WHERE e.deptno = a.deptno**

**AND hiredate BETWEEN TO\_DATE('1981/01/01','YYYY/MM/DD')**

**AND TO\_DATE('1981/12/31','YYYY/MM/DD')**

**ORDER BY e.deptno ;**

**답안 3. 분석 함수 이용**

**SQL> SELECT empno, ename, sal, hiredate, deptno,**

**ROUND(AVG(sal) OVER(PARTITION BY deptno),2) AS AVG\_SAL**

**FROM emp**

**WHERE hiredate BETWEEN TO\_DATE('1981/01/01','YYYY/MM/DD')**

**AND TO\_DATE('1981/12/31','YYYY/MM/DD') ;**

EMPNO ENAME SAL HIREDATE DEPTNO AVG\_SAL

---------- ---------- ---------- -------- ---------- ----------

7782 CLARK 2450 81/06/09 10 3725

7839 KING 5000 81/11/17 10 3725

7566 JONES 2975 81/04/02 20 2987.5

7902 FORD 3000 81/12/03 20 2987.5

7698 BLAKE 2850 81/05/01 30 1566.67

7844 TURNER 1500 81/09/08 30 1566.67

7654 MARTIN 1250 81/09/28 30 1566.67

7900 JAMES 950 81/12/03 30 1566.67

7521 WARD 1250 81/02/22 30 1566.67

7499 ALLEN 1600 81/02/20 30 1566.67

10 rows selected.

분석 함수는 WHERE절의 조건식 비교 후 결과 값을 생성한다. 때문에 위와 같은 문장은 조건절이 먼저 비교되면서 부서별 평균 급여는 잘못된 값을 보여줄 수 있다.

**SQL> SELECT empno, ename, sal, hiredate, deptno, avg\_sal**

**FROM (SELECT empno, ename, sal, hiredate, deptno,**

**ROUND(AVG(sal) OVER(PARTITION BY deptno),2) AS AVG\_SAL**

**FROM emp)**

**WHERE hiredate BETWEEN TO\_DATE('1981/01/01','YYYY/MM/DD')**

**AND TO\_DATE('1981/12/31','YYYY/MM/DD') ;**

1. EMP 테이블에서 사원들의 정보를 EMPNO 컬럼으로 정렬하고, 각 사원의 급여를 행 별로 누적하여 TOTAL 컬럼을 검색하시오.

검색 결과

EMPNO ENAME SAL TOTAL

---------- ---------- ---------- ----------

7369 SMITH 800 800

7499 ALLEN 1600 2400

7521 WARD 1250 3650

7566 JONES 2975 6625

7654 MARTIN 1250 7875

7698 BLAKE 2850 10725

7782 CLARK 2450 13175

7788 SCOTT 3000 16175

7839 KING 5000 21175

7844 TURNER 1500 22675

7876 ADAMS 1100 23775

7900 JAMES 950 24725

7902 FORD 3000 27725

7934 MILLER 1300 29025

14 rows selected.

**답안 1. Correlated Subquery 이용**

**SQL> SELECT a.empno, a.ename, a.sal, (SELECT SUM(sal)**

**FROM emp**

**WHERE empno <= a.empno) AS TOTAL**

**FROM emp a**

**ORDER BY a.empno ;**

**답안 2. Self Join 이용**

**SQL> SELECT a.empno, a.ename, a.sal , SUM(b.sal) AS TOTAL**

**FROM emp a , emp b**

**WHERE a.empno >= b.empno**

**GROUP BY a.empno, a.ename, a.sal**

**ORDER BY a.empno ;**

**답안 3. 분석 함수 이용**

**SQL> SELECT empno, ename, sal,**

**SUM(sal) OVER(ORDER BY empno ROWS BETWEEN UNBOUNDED PRECEDING**

**AND CURRENT ROW) AS TOTAL**

**FROM emp ;**

JOIN, Subquery 등을 이용하여도 결과는 생성 가능하지만 동일 집합의 반복 접근은 I/O를 증가시키며 성능을 악화 시킬 수 있다. 필요에 따라 Join, Subquery를 이용해야 할 수도 있지만 분석 함수의 사용을 고려한다.

**추가 실습**

**SQL> SELECT empno, ename, sal, SUM(sal) OVER (ORDER BY empno) AS TOTAL**

**FROM emp ;**

EMPNO ENAME SAL TOTAL

---------- ---------- ---------- ----------

7369 SMITH 800 800

7499 ALLEN 1600 2400

7521 WARD 1250 3650

7566 JONES 2975 6625

7654 MARTIN 1250 7875

7698 BLAKE 2850 10725

7782 CLARK 2450 13175

...

14 rows selected.

**SQL> SELECT empno, ename, sal,**

**SUM(sal) OVER(ORDER BY empno ROWS BETWEEN CURRENT ROW**

**AND UNBOUNDED FOLLOWING) AS TOTAL**

**FROM emp ;**

EMPNO ENAME SAL TOTAL

---------- ---------- ---------- ----------

7369 SMITH 800 29025

7499 ALLEN 1600 28225

7521 WARD 1250 26625

7566 JONES 2975 25375

7654 MARTIN 1250 22400

7698 BLAKE 2850 21150

7782 CLARK 2450 18300

...

14 rows selected.

**SQL> SELECT empno, ename, sal,**

**SUM(sal) OVER(ORDER BY empno ROWS BETWEEN 1 PRECEDING**

**AND 1 FOLLOWING) AS TOTAL**

**FROM emp ;**

EMPNO ENAME SAL TOTAL

---------- ---------- ---------- ----------

7369 SMITH 800 2400

7499 ALLEN 1600 3650

7521 WARD 1250 5825

7566 JONES 2975 5475

7654 MARTIN 1250 7075

7698 BLAKE 2850 6550

7782 CLARK 2450 8300

7788 SCOTT 3000 10450

7839 KING 5000 9500

7844 TURNER 1500 7600

7876 ADAMS 1100 3550

7900 JAMES 950 5050

7902 FORD 3000 5250

7934 MILLER 1300 4300

14 rows selected.

1. **EMP 테이블에서 HIREDATE, EMPNO 컬럼으로 정렬된 사원 정보를 검색하시오.**

**이때 해당 사원보다 위, 아래(앞, 뒤)의 입사 일자를 함께 출력합니다.**

**검색 결과**

EMPNO ENAME HIREDATE PREV\_HIRE NEXT\_HIRE

---------- ---------- --------------- --------------- ---------------

7369 SMITH 80/12/17 81/02/20

7499 ALLEN 81/02/20 80/12/17 81/02/22

7521 WARD 81/02/22 81/02/20 81/04/02

7566 JONES 81/04/02 81/02/22 81/05/01

7698 BLAKE 81/05/01 81/04/02 81/06/09

7782 CLARK 81/06/09 81/05/01 81/09/08

7844 TURNER 81/09/08 81/06/09 81/09/28

7654 MARTIN 81/09/28 81/09/08 81/11/17

7839 KING 81/11/17 81/09/28 81/12/03

7900 JAMES 81/12/03 81/11/17 81/12/03

7902 FORD 81/12/03 81/12/03 82/01/23

7934 MILLER 82/01/23 81/12/03 87/04/19

7788 SCOTT 87/04/19 82/01/23 87/05/23

7876 ADAMS 87/05/23 87/04/19

14 rows selected.

**답안 1. ROWNUM을 이용한 Outer Join 사용**

**SQL> SELECT a.empno, a.ename, a.hiredate,b.hiredate AS PREV\_HIRE,c.hiredate AS NEXT\_HIRE**

**FROM (SELECT ROWNUM no1, empno, ename, hiredate**

**FROM (SELECT empno, ename, hiredate**

**FROM emp**

**ORDER BY hiredate,empno)) a,**

**(SELECT ROWNUM+1 no2, empno, ename, hiredate**

**FROM (SELECT empno, ename, hiredate**

**FROM emp**

**ORDER BY hiredate,empno)) b,**

**(SELECT ROWNUM-1 no3, empno, ename, hiredate**

**FROM (SELECT empno, ename, hiredate**

**FROM emp**

**ORDER BY hiredate,empno)) c**

**WHERE a.no1 = b.no2 (+)**

**AND a.no1 = c.no3 (+)**

**ORDER BY a.hiredate, a.empno ;**

**답안 2. 분석 함수 이용**

**SQL> SELECT empno, ename, hiredate,**

**LAG(hiredate) OVER (ORDER BY hiredate,empno) AS PREV\_HIRE,**

**LEAD(hiredate) OVER (ORDER BY hiredate,empno) AS NEXT\_HIRE**

**FROM emp ;**

***함수 소개: LAG / LEAD***

지정된 개수의 이전, 이후 행의 값 가져오기. WINDOWING 절을 지정하지 못하며 NULL 값을 대체하는 값을 지정할 수 있음. (NVL 불필요)

**Q. 30번 부서의 사원을 이름순으로 정렬하여 검색하며 이전, 다음 행의 급여를 함께 표시**

**SQL> SELECT empno, ename, sal,**

**LAG (sal,1,0) over ( order by ename ) prev\_sal,**

**LEAD (sal,1,0) over ( order by ename ) next\_sal**

**FROM emp**

**WHERE deptno = 30 ;**

EMPNO ENAME SAL PREV\_SAL NEXT\_SAL

---------- ---------- ---------- ---------- ----------

7499 ALLEN 1600 0 2850

7698 BLAKE 2850 1600 950

7900 JAMES 950 2850 1250

7654 MARTIN 1250 950 1500

7844 TURNER 1500 1250 1250

7521 WARD 1250 1500 0

6 rows selected.

1. **EMP 테이블에서 다음과 같이 부서별 사원 이름을 검색하시오.**

**검색 결과**

DEPTNO EMPLOYEES

---------- --------------------------------------------------

10 CLARK,KING,MILLER

20 ADAMS,FORD,JONES,SCOTT,SMITH

30 ALLEN,BLAKE,JAMES,MARTIN,TURNER,WARD

**답안 1. LISTAGG 사용 (v11gR2)**

**SQL> SELECT deptno,**

**LISTAGG(ename,',') WITHIN GROUP (ORDER BY ename) AS employee**

**FROM emp**

**GROUP BY deptno ;**

DEPTNO EMPLOYEE

---------- ----------------------------------------

10 CLARK,KING,MILLER

20 ADAMS,FORD,JONES,SCOTT,SMITH

30 ALLEN,BLAKE,JAMES,MARTIN,TURNER,WARD

**추가 실습**

**SQL> SELECT d.dname, LISTAGG(e.ename||'('||e.sal||')',',')**

**WITHIN GROUP (ORDER BY ename) AS employee**

**FROM emp e, dept d**

**WHERE e.deptno = d.deptno**

**GROUP BY d.dname ;**

DNAME EMPLOYEE

-------------- -------------------------------------------------------------------------

ACCOUNTING CLARK(2450),KING(5000),MILLER(1300)

RESEARCH ADAMS(1100),FORD(3000),JONES(2975),SCOTT(3000),SMITH(800)

SALES ALLEN(1600),BLAKE(2850),JAMES(950),MARTIN(1250),TURNER(1500),WARD(1250)

3 rows selected.

1. **EMP 테이블에서 DEPTNO, SAL, ENAME 컬럼을 기준으로 정렬된 정보를 검색하면서 부서별 누적된 급여, 부서별 급여의 백분율, 전체 사원의 급여 합계에서의 백분율을 검색한다.**

**검색 결과**

DEPTNO ENAME SAL CUM\_SAL PCT\_DEPT PCT\_OVERALL

---------- ---------- ---------- ---------- ---------- -----------

10 MILLER 1300 1300 14.9 4.5

10 CLARK 2450 3750 28 8.4

10 KING 5000 8750 57.1 17.2

20 SMITH 800 800 7.4 2.8

20 ADAMS 1100 1900 10.1 3.8

20 JONES 2975 4875 27.4 10.2

20 FORD 3000 7875 27.6 10.3

20 SCOTT 3000 10875 27.6 10.3

30 JAMES 950 950 10.1 3.3

30 MARTIN 1250 2200 13.3 4.3

30 WARD 1250 3450 13.3 4.3

30 TURNER 1500 4950 16 5.2

30 ALLEN 1600 6550 17 5.5

30 BLAKE 2850 9400 30.3 9.8

14 rows selected.

**답안 1. Subquery 이용**

**SQL> SELECT e1.deptno, e1.ename, e1.sal,**

**SUM(e4.sal) AS cum\_sal,**

**ROUND(100\*e1.sal/e2.sal\_by\_dept,1) AS pct\_dept,**

**ROUND(100\*e1.sal/e3.sal\_overall,1) AS pct\_overall**

**FROM emp e1,**

**(SELECT deptno, SUM(sal) sal\_by\_dept**

**FROM emp**

**GROUP BY deptno) e2 ,**

**(SELECT SUM(sal) sal\_overall**

**FROM emp) e3,**

**emp e4**

**WHERE e1.deptno = e2.deptno**

**AND e1.deptno = e4.deptno**

**AND (e1.sal > e4.sal OR (e1.sal = e4.sal AND e1.ename >= e4.ename))**

**GROUP BY e1.deptno, e1.ename, e1.sal,**

**ROUND(100\*e1.sal/e2.sal\_by\_dept,1) ,**

**ROUND(100\*e1.sal/e3.sal\_overall,1)**

**ORDER BY deptno,sal,ename ;**

**답안 2. 분석 함수 이용**

**SQL> SELECT deptno, ename, sal,**

**SUM(sal) OVER(PARTITION BY deptno ORDER BY sal, ename) AS cum\_sal,**

**ROUND(100\*RATIO\_TO\_REPORT(sal) OVER(PARTITION BY deptno),1) AS pct\_dept,**

**ROUND(100\*RATIO\_TO\_REPORT(sal) OVER(),1) AS pct\_overall**

**FROM emp ;**

***함수 소개 : RATIO\_TO\_REPORT***

: WINDOW 영역의 합계 내에서 현재 값이 차지하는 백분율. 별도의 WINDOWING 절을 설정하는 것은 불가능하다.