이 쿼리들은 데이터베이스의 테이블과 뷰(View) 생성, 데이터 조작(삽입, 수정, 삭제), 그리고 집계 및 요약 정보를 다룹니다. 각 쿼리에 대한 설명을 추가하겠습니다.

---

1. 부서 정보 확인

```
SELECT * FROM department;
```

- `SELECT *`: `department` 테이블에서 모든 컬럼을 조회. 부서 정보가 담겨 있는 테이블을 확인하는 쿼리입니다.

2. 부서 추가

```
INSERT INTO department (dept_no, dept_name)
VALUES ('A5', '전산부');
```

- `INSERT INTO`: `department` 테이블에 새로운 부서를 추가합니다. `dept_no`는 'A5', `dept_name`은 '전산부'로 지정됩니다.

3. 제품 정보 추가

```
INSERT INTO product (product_id, product_name, product_unit, unit_price,
stock)
VALUES (91, '연어소스', '박스', 5000, 40);
```

- `INSERT INTO`: `product` 테이블에 새로운 제품 정보를 추가합니다. 제품 ID 91번, '연어소스'라는 제품명, '박스' 단위, 5000원 가격, 재고 40개를 추가합니다.

4. 사원 정보 확인

```
SELECT * FROM employee;
```

- `SELECT *`: `employee` 테이블에서 모든 사원 정보를 조회합니다.

5. 사원 정보 추가

```
INSERT INTO employee (emp_no, name, eng_name, position, gender,
birth_date, hire_date, address, city, region, home_phone, manager_no,
dept_id)
VALUES
```

```
('2024120101', '이미애', 'Lee Mi Ae', '부장', '남', '1972-01-15', '2024-12-
01', '서울시 양천구 목동 123', '서울', '서울특별시', '02-2644-4567', NULL, NULL),
('2024120102', '박승진', 'Park seng jin', '부장', '여', '1978-06-20', '2024-
12-10', '서울시 양천구 목동 456', '서울', '서울특별시', '02-2645-6543', NULL,
NULL),
('2024120103', '김지훈', 'Kim Jihoon', '부장', '여', '1971-11-30', '2024-12-
01', '서울시 양천구 목동 789', '서울', '서울특별시', '02-2643-4567', NULL, NULL);
```

- `INSERT INTO` : `employee` 테이블에 새로운 사원 정보를 추가합니다. 여러 명의 사원 정보를 한 번에 추가합니다.

6. 사원 정보 업데이트

```
UPDATE employee
SET name = '이은숙'
WHERE emp_id = 47;
```

- `UPDATE` : 사원 ID가 47인 사원의 이름을 '이은숙'으로 업데이트합니다.

7. 제품 정보 확인

```
SELECT * FROM product;
```

- `SELECT *` : `product` 테이블에서 모든 제품 정보를 조회합니다.

8. 제품 정보 업데이트 (단위 변경)

```
UPDATE product
SET product_unit = '병'
WHERE product_id = 91;
```

- `UPDATE` : 제품 ID가 91인 제품의 단위를 '박스'에서 '병'으로 변경합니다.

9. 제품 가격과 재고 업데이트

```
UPDATE product
SET unit_price = unit_price * 1.1,
    stock = stock - 10
WHERE product_id = 91;
```

- `UPDATE` : 제품 가격을 10% 인상하고, 재고를 10개 줄입니다.

10. 제품 정보 삭제

```sql
DELETE FROM product
WHERE product_id = 91;
```

- `DELETE` : 제품 ID가 91인 제품 정보를 삭제합니다.

11. 사원 정보 삭제 (최근 고용된 3명)

```sql
DELETE FROM employee
WHERE emp_no IN (
    SELECT emp_no
    FROM employee
    ORDER BY hire_date DESC
    LIMIT 3
);
```

- `DELETE` : `employee` 테이블에서 최근 3명 고용된 사원의 정보를 삭제합니다. 서브쿼리로 `hire_date` 가 최신인 사원들을 조회하여 삭제합니다.

12. 제품 정보 추가 또는 업데이트 (ON CONFLICT 사용)

```sql
INSERT OR REPLACE INTO product (product_id, product_name, unit_price,
stock)
VALUES (92, '불닭볶음면', 6000, 50);
```

- `INSERT OR REPLACE` : 제품 ID가 92번인 제품이 이미 존재하면 기존 정보를 업데이트하고, 없으면 새로 추가합니다. `ON CONFLICT` 는 `INSERT` 에만 적용됩니다.

13. 고객 주문 요약 테이블 생성

```sql
CREATE TABLE IF NOT EXISTS customer_order_summary (
    customer_id CHAR(5) PRIMARY KEY,
    company_name VARCHAR(50),
    order_count INT,
    last_order_date DATE
);
```

- `CREATE TABLE` : 고객별 주문 요약을 위한 테이블을 생성합니다. 고객 ID, 회사명, 주문 건수, 마지막 주문 날짜를 저장합니다.

14. 고객 주문 요약 테이블 정보 추가 (INSERT SELECT)

```sql
INSERT INTO customer_order_summary (customer_id, company_name,
order_count, last_order_date)
SELECT c.cust_id, c.company_name, COUNT(*), MAX(o.order_date)
```

```
FROM customer c
INNER JOIN orders o ON c.cust_id = o.cust_id
GROUP BY c.cust_id, c.company_name;
```

- `INSERT INTO ... SELECT` : 고객의 주문 건수와 마지막 주문 날짜를 `customer_order_summary` 테이블에 추가합니다. `INNER JOIN` 을 사용하여 고객과 주문 정보를 연결하고, `GROUP BY` 로 고객별로 요약합니다.

15. 제품 가격을 소스 가격의 평균으로 업데이트

```
UPDATE product
SET unit_price = (SELECT AVG(unit_price) FROM product WHERE product_name
LIKE '%소스%')
WHERE product_id = 91;
```

- `UPDATE` : 제품 ID 91번의 가격을 제품 이름에 '소스'가 포함된 제품들의 평균 가격으로 업데이트합니다. 서브쿼리를 사용하여 평균 가격을 계산합니다.

16. 주문 정보가 있는 고객의 마일리지 10% 증가

```
UPDATE customer
SET mileage = mileage * 1.1
WHERE cust_id IN (
    SELECT DISTINCT cust_id
    FROM orders
);
```

- `UPDATE` : 주문을 한 고객들의 마일리지를 10% 증가시킵니다. 서브쿼리로 `orders` 테이블에서 주문한 고객을 조회합니다.

17. VIP 고객 마일리지 1000점 추가

```
UPDATE customer
SET mileage = mileage + 1000
WHERE cust_id IN (
    SELECT cust_id
    FROM mileage_grade mg
    WHERE mileage BETWEEN mg.min_mileage AND mg.max_mileage
      AND mg.grade = 'VIP'
);
```

- `UPDATE` : 마일리지 등급이 'VIP'인 고객들에게 마일리지 1000점을 추가합니다. `mileage_grade` 테이블을 조인하여 VIP 등급에 해당하는 고객을 조회합니다.

18. 주문 상세에 없는 주문 번호 삭제

```
DELETE FROM orders
WHERE order_id NOT IN (
    SELECT DISTINCT order_id
    FROM order_details
);
```

- `DELETE` : 주문 상세(`order_details`)에 해당하지 않는 주문 번호를 가진 주문을 삭제합니다. 서브쿼리를 사용하여 존재하지 않는 주문을 찾아 삭제합니다.

19. 주문 정보가 없는 고객 삭제

```
DELETE FROM customer
WHERE cust_id NOT IN (
    SELECT DISTINCT cust_id
    FROM orders
);
```

- `DELETE` : 주문이 없는 고객을 삭제합니다. `orders` 테이블에서 주문이 없는 고객을 서브쿼리로 찾습니다.

20. 사원 뷰 생성: 이름, 전화번호, 입사일, 주소

```
CREATE VIEW IF NOT EXISTS view_employee AS
SELECT name AS 이름,
       home_phone AS 전화번호,
       hire_date AS 입사일,
       address AS 주소
FROM employee;
```

- `CREATE VIEW` : 사원 정보를 보기 쉽게 표시하기 위해 뷰를 생성합니다. 뷰에는 이름, 전화번호, 입사일, 주소가 포함됩니다.

21. 사원 뷰 확인

```
SELECT * FROM view_employee;
```

- `SELECT *` : `view_employee` 뷰에서 사원 정보를 조회합니다.