이 쿼리들은 SQL에서 데이터 조작 언어(DML)와 데이터 정의 언어(DDL)를 사용하여 데이터를 관리하는 다양한 작업을 수행하는 예시들입니다. 각 쿼리의 목적과 동작을 하나씩 설명하겠습니다.

1. 부서 정보 확인

```
SELECT * FROM department d;
```

- **설명:** `department` 테이블에 있는 모든 부서 정보를 조회합니다.

2. 부서 추가

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

- **설명:** `department` 테이블에 새로운 부서인 '전산부'를 추가합니다. 부서 번호는 'A5'입니다.

3. 제품 정보 추가

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

- **설명:** `product` 테이블에 '연어소스'라는 새로운 제품을 추가합니다. 제품 ID는 91번, 단위는 '박스', 가격은 5000원, 재고는 40입니다.

4. 사원 정보 확인

```
SELECT * FROM employee e;
```

- **설명:** `employee` 테이블의 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);
```

- **설명:** `employee` 테이블에 세 명의 새로운 사원을 추가합니다. 각 사원에 대한 상세 정보(사원 번호, 이름, 영문 이름, 직위, 성별, 생년월일, 입사일, 주소 등)가 포함됩니다.

6. 사원 정보 업데이트

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

- **설명:** `employee` 테이블에서 사원 ID가 47번인 사원의 이름을 '이은숙'으로 업데이트합니다.

7. 제품 정보 조회

```
SELECT * FROM product;
```

- **설명:** `product` 테이블의 모든 제품 정보를 조회합니다.

8. 제품 정보 업데이트 (단위 변경)

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

- **설명:** 제품 ID가 91번인 제품의 단위를 '병'으로 변경합니다.

9. 제품 정보 업데이트 (가격 및 재고 변경)

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

- **설명:** 제품 ID가 91번인 제품의 가격을 10% 인상하고, 재고를 10개 줄입니다.

10. 제품 정보 삭제

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

- **설명:** 제품 ID가 91번인 제품을 `product` 테이블에서 삭제합니다.

11. 최근에 고용된 사원 3명 삭제

```
DELETE FROM employee
JOIN (
    SELECT emp_id
    FROM employee
    ORDER BY hire_date DESC
    LIMIT 3
) AS recent_employees
ON employee.emp_id = recent_employees.emp_id;
```

- **설명:** 최근에 고용된 사원 3명을 `employee` 테이블에서 삭제합니다. 서브쿼리에서 최근 고용된 사원 ID를 가져와 삭제합니다.

12. 제품 정보 추가 또는 업데이트

```
INSERT INTO product (product_id, product_name, unit_price, stock)
VALUES (92, '불닭볶음면', 6000, 50)
ON CONFLICT (product_id)
DO UPDATE SET
    product_name = EXCLUDED.product_name,
    unit_price = EXCLUDED.unit_price,
    stock = EXCLUDED.stock;
```

- **설명:** 제품 ID가 92번인 제품을 `product` 테이블에 추가합니다. 만약 제품 ID가 이미 존재한다면, 해당 제품의 정보를 업데이트합니다.

13. 고객 주문 요약 테이블 생성

```
CREATE TABLE customer_order_summary (
    customer_id CHAR(5) PRIMARY KEY,
    company_name VARCHAR(50),
```

```
    order_count INT,
    last_order_date DATE
);
```

- **설명:** `customer_order_summary`라는 고객 주문 요약 테이블을 생성합니다. 이 테이블에는 고객 ID, 회사 이름, 주문 건수, 마지막 주문 날짜가 포함됩니다.

14. 고객 주문 요약 정보 추가

```
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;
```

- **설명:** 고객과 주문 정보를 조인하여 각 고객의 주문 건수와 마지막 주문 날짜를 계산하여 `customer_order_summary` 테이블에 삽입합니다.

15. 제품 가격을 소스 제품의 평균 가격으로 업데이트

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

- **설명:** 제품 ID가 91번인 제품의 가격을 '소스'라는 이름이 포함된 모든 제품의 평균 가격으로 업데이트합니다.

16. 주문이 있는 사용자의 마일리지 10% 증가

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

- **설명:** 주문 기록이 있는 고객들의 마일리지를 10% 증가시킵니다.

17. VIP 고객에게 마일리지 1000점 추가

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

- **설명:** VIP 등급에 해당하는 고객들에게 마일리지 1000점을 추가합니다.

18. 주문 상세에 없는 주문 번호 삭제

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

- **설명:** `order_details` 테이블에 존재하지 않는 주문 번호를 가진 `orders` 테이블의 주문을 삭제합니다.

19. 주문 정보가 없는 고객 삭제

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

- **설명:** `orders` 테이블에 주문 정보가 없는 고객을 `customer` 테이블에서 삭제합니다.

20. 사원 뷰 생성: 이름, 전화번호, 입사일, 주소

```
CREATE OR REPLACE VIEW view_employee AS
SELECT name AS 이름, home_phone AS 전화번호, hire_date AS 입사일, address AS 주소
FROM employee;
```

- **설명:** 사원의 이름, 전화번호, 입사일, 주소를 포함하는 `view_employee` 라는 뷰(View)를 생성합니다.

21. 제품별 주문 수량 합계 뷰 생성

```
CREATE OR REPLACE VIEW view_product_order_quantity_sum AS
SELECT p.product_name AS 제품명, SUM(o.quantity) AS 주문수량합
FROM product p
INNER JOIN order_details o ON p.product_id = o.product_id
GROUP BY p.product_name;
```

- **설명:** 제품별 주문 수량의 합계를 계산하여 `view_product_order_quantity_sum`이라는 뷰를 생성합니다.

22. 여성 사원 정보를 담은 뷰 생성

```
CREATE OR REPLACE VIEW view_female_employees AS
SELECT name AS 이름, home_phone AS 전화번호, hire_date AS 입사일, address AS 주
소, gender AS 성별
FROM employee
WHERE gender = '여';
```

- **설명:** 여성 사원의 정보를 포함하는 `view_female_employees`라는 뷰를 생성합니다.

이와 같은 쿼리들을 통해 SQL을 사용하여 데이터베이스 내에서 데이터를 조작하고 구조를 관리할 수 있습니다.