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.
이 쿼리들은 다양한 JOIN 유형과 서브쿼리, 집계 함수를 사용해 데이터를 분석하고 조합하는 방법을 보여줍니다. 각각의 쿼리에 대한 설명을 추가하겠습니다.

---

1. **사원 테이블과 부서 테이블을 조인해서 사원명 '홍길동'의 부서번호, 부서명, 사원명, 사원번호 출력**

```sql
SELECT d.dept_id,
       d.dept_name,
       e.name,
       e.dept_id
FROM department d
JOIN employee e
ON d.dept_id = e.dept_id
WHERE e.name = '홍길동';
```

   - `JOIN` : 부서(`department`)와 사원(`employee`) 테이블을 부서 번호(`dept_id`)를 기준으로 조인.
   - 조건으로 사원 이름이 '홍길동'인 행을 필터링하여 해당 사원의 부서와 관련된 정보를 출력.
2. **고객과 주문 테이블을 조인하여 고객명, 회사명, 주문건수를 출력 (주문건수 많은 순으로 정렬)**

```sql
SELECT c.cust_id,
       c.contact_name,
       c.company_name,
       COUNT(*) AS order_count
FROM customer c
JOIN orders o
ON c.cust_id = o.cust_id
GROUP BY c.cust_id,
         c.contact_name,
         c.company_name
ORDER BY order_count DESC;
```

   - `JOIN` : 고객 테이블(`customer`)과 주문 테이블(`orders`)을 고객 아이디(`cust_id`)로 조인.
   - 각 고객의 주문 건수를 집계하고, 주문 건수가 많은 순으로 정렬.
3. **고객, 주문, 주문 상세 테이블을 조인하여 고객 아이디, 담당자명, 회사명, 주문 총액 출력 (총액 순으로 정렬)**

```sql
SELECT c.cust_id,
       c.contact_name,
```

```
       c.company_name,
       SUM(od.quantity * od.unit_price) AS total_order_amount
FROM customer c
JOIN orders o
ON c.cust_id = o.cust_id
JOIN order_details od
ON o.order_id = od.order_id
GROUP BY c.cust_id,
         c.contact_name,
         c.company_name
ORDER BY total_order_amount DESC;
```

- 세 개의 테이블을 조인하여 각 고객의 주문 총액을 계산.
- `SUM(od.quantity * od.unit_price)` : 주문 항목별 수량과 단가를 곱해 주문 총액을 구함.
- 주문 총액 순으로 결과를 정렬.

4. **고객과 마일리지 등급 테이블을 조인하여 고객 아이디, 회사명, 담당자명, 마일리지, 등급 출력 (조건: 담당자명 '남성우')**

```
SELECT c.cust_id,
       c.company_name,
       c.contact_name,
       c.mileage,
       mg.grade
FROM customer c
JOIN mileage_grade mg
ON c.mileage BETWEEN mg.min_mileage AND mg.max_mileage
WHERE c.contact_name = '남성우';
```

- 고객의 마일리지가 등급 범위 내에 포함되면 등급 정보를 매칭하여 출력.
- `BETWEEN` : 마일리지가 등급 테이블의 최소 및 최대 범위 내에 있는 고객을 필터링.

5. **사원 테이블과 부서 테이블을 외부 조인하여 부서명이 없는 경우 NULL로 출력**

```
SELECT e.emp_id,
       e.name,
       d.dept_name
FROM employee e
LEFT JOIN department d
ON e.dept_id = d.dept_id;
```

- `LEFT JOIN` : 사원 테이블의 모든 행을 반환하고, 부서 정보가 없는 경우 `NULL` 로 출력.
- 부서가 없는 사원에 대해 부서명이 `NULL` 로 표시됨.

6. **사원 테이블을 조인하여 사원명, 부서장명 출력 (부서장이 없으면 NULL)**

```sql
SELECT e.name AS name,
       e.position,
       supervisor.name AS supervisor_name
FROM employee e
LEFT JOIN employee supervisor
ON e.manager_no = supervisor.emp_id
ORDER BY supervisor_name;
```

- `LEFT JOIN`: 사원 테이블을 자기 자신과 조인하여 매니저(부서장)의 정보를 연결.
- 매니저 번호(`manager_no`)를 기준으로 상사 이름을 조회하며, 상사가 없을 경우 `NULL`이 출력됨.

7. **최대 마일리지를 가진 고객의 정보 출력**

```sql
SELECT cust_id,
       company_name,
       contact_name,
       mileage
FROM customer
WHERE mileage = (SELECT MAX(mileage) FROM customer);
```

- 서브쿼리를 사용하여 최대 마일리지를 가진 고객을 필터링.
- 최대 마일리지를 가진 고객의 ID, 회사명, 담당자명, 마일리지 출력.

8. **주문 번호가 13번인 고객의 정보 출력**

```sql
SELECT c.company_name,
       c.contact_name
FROM customer c
JOIN orders o
ON c.cust_id = o.cust_id
WHERE o.order_id = 13;
```

- 주문 번호가 13번인 주문에 대한 고객 정보를 출력.

9. **부산에 있는 고객 중에서 최소 마일리지보다 큰 고객 정보 출력**

```sql
SELECT contact_name,
       company_name,
       mileage
FROM customer
WHERE mileage > (SELECT MIN(mileage) FROM customer WHERE city LIKE '%
부산%');
```

- 부산에 있는 고객 중에서 최소 마일리지보다 큰 고객을 필터링.

10. **도시별 평균 마일리지를 구하는데, 고객의 평균 마일리지보다 큰 경우만 출력**

```
SELECT city,
       AVG(mileage) AS avg_mileage
FROM customer
GROUP BY city
HAVING avg_mileage > (SELECT AVG(mileage) FROM customer);
```

- 각 도시별 평균 마일리지를 계산한 후, 전체 고객의 평균 마일리지보다 큰 도시만 출력.

11. **도시별 평균 마일리지와 고객의 마일리지 차이를 구하는 쿼리**

```
WITH city_summary AS (
    SELECT city,
           AVG(mileage) AS avg_mileage
    FROM customer
    GROUP BY city
)
SELECT c.contact_name,
       c.company_name,
       c.mileage,
       c.city,
       cs.avg_mileage,
       cs.avg_mileage - c.mileage AS difference
FROM customer c
JOIN city_summary cs
ON c.city = cs.city;
```

- `WITH` 절을 사용해 도시별 평균 마일리지를 구한 결과를 임시 테이블로 저장하고, 고객별 마일리지와 평균 마일리지의 차이를 계산.

12. **고객별로 가장 최근 주문 일자를 구하는 쿼리**

```
SELECT cust_id,
       contact_name,
       (SELECT MAX(order_date)
        FROM orders
        WHERE orders.cust_id = customer.cust_id) AS last_order_date
FROM customer;
```

- 서브쿼리를 사용하여 고객별로 가장 최근 주문 일자를 반환.

13. **사원 테이블에서 사원번호, 사원명, 매니저 번호, 매니저 이름 출력**

```
SELECT emp_id,
       name,
       manager_no,
       (SELECT name
        FROM employee AS supervisor
```

```
        WHERE supervisor.emp_id = e.manager_no) AS supervisor_name
FROM employee e;
```

- 서브쿼리를 사용하여 매니저의 이름을 출력. 매니저 번호를 이용해 상사 이름을 조회.

14. **도시 별 최대 마일리지를 가진 고객 정보 출력**

```
SELECT city,
       contact_name,
       company_name,
       mileage
FROM customer
WHERE (city, mileage) IN (
    SELECT city, MAX(mileage)
    FROM customer
    GROUP BY city
);
```

- 서브쿼리를 사용하여 각 도시별 최대 마일리지를 가진 고객을 필터링하여 출력.

이 쿼리들은 다양한 조인과 서브쿼리, 그룹화, 집계 함수를 활용하여 데이터를 조합하고 분석하는 방법을 보여줍니다.