1. 고객 전체를 검색하세요

```
SELECT * FROM customer;
```

- **설명**: 고객 테이블에 있는 모든 행을 출력합니다. `SELECT *`는 모든 열을 의미합니다. 결과는 테이블의 전체 데이터를 반환합니다.

2. 고객의 마일리지 10% 추가된 값 출력

```
SELECT cust_id, contact_name, company_name, mileage AS points, mileage *
1.1 AS "10%추가"
FROM customer;
```

- **설명**: 고객 코드, 연락처, 회사명, 마일리지를 출력하면서 마일리지에 10%를 더한 값을 계산해서 추가로 출력합니다. `mileage * 1.1`로 10% 추가된 마일리지를 계산하고 별칭을 부여했습니다.

3. 마일리지가 10000점 이상인 고객 검색

```
SELECT cust_id, contact_name, mileage FROM customer WHERE mileage >=
10000;
```

- **설명**: `WHERE` 절을 사용하여 마일리지가 10000점 이상인 고객만 검색합니다. 조건을 만족하는 고객의 코드, 이름, 마일리지만 출력합니다.

4. 도시가 서울인 고객을 마일리지 역순으로 출력

```
SELECT cust_id, contact_name, city, mileage AS points FROM customer WHERE
city = '서울' ORDER BY mileage DESC;
```

- **설명**: 도시가 '서울'인 고객만 검색하며, 마일리지 역순으로 정렬하여 출력합니다. `ORDER BY mileage DESC`는 높은 순서대로 정렬하라는 의미입니다.

5. 고객 테이블에서 3명만 출력

```
SELECT * FROM customer LIMIT 3;
```

- **설명**: `LIMIT 3`는 첫 번째부터 세 번째까지의 행만 반환합니다. 주로 테이블의 일부분만 보고 싶을 때 사용됩니다.

6. 마일리지 역순으로 3명만 출력

```sql
SELECT * FROM customer ORDER BY mileage DESC LIMIT 3;
```

- **설명**: 고객을 마일리지 기준으로 역순으로 정렬하고, 그중 상위 3명만 출력합니다. `LIMIT 3`과 정렬 기준을 결합하여 사용합니다.

7. 도시 이름만 중복 없이 출력

```sql
SELECT DISTINCT city FROM customer;
```

- **설명**: `DISTINCT`는 중복되는 값을 제거하고 고유한 값만 출력합니다. 여기서는 도시 이름이 중복 없이 출력됩니다.

8. 산술 연산자 사용

```sql
SELECT 33 + 5 AS addition, 33 - 5 AS subtraction, 33 * 5 AS
multiplication, CAST(33 AS FLOAT) / 5 AS float_division, 33 / 5 AS
integer_division, 33 % 5 AS remainder1;
```

- **설명**: 다양한 산술 연산을 수행합니다. `CAST(33 AS FLOAT)`로 33을 실수형으로 변환하여 소수점 연산이 가능하도록 했습니다. `%`는 나머지 연산자입니다.

9. 비교 연산자 사용

```sql
SELECT 33 >= 5 AS "greater_than_or_equal", 33 <= 5 AS
"less_than_or_equal", 43 > 23 AS "greater_than", 43 < 23 AS "less_than",
43 = 43 AS "equal", 43 != 23 AS "not_equal";
```

- **설명**: 다양한 비교 연산을 수행합니다. 결과는 `TRUE` 또는 `FALSE`로 출력됩니다.

10. 직위가 대표가 아닌 고객 검색

```sql
SELECT * FROM customer WHERE contact_position != '대표';
```

- **설명**: 직위가 '대표'가 아닌 고객을 출력합니다. `!=`는 '같지 않다'라는 의미입니다.

11. 부산이고 마일리지가 1000 이하인 고객 검색

```sql
SELECT * FROM customer WHERE city = '부산' AND mileage <= 1000;
```

- **설명**: `AND` 조건을 사용하여 도시가 '부산'이면서 동시에 마일리지가 1000 이하인 고객만 출력합니다.

12. 도시가 부산인 사람과 마일리지가 1000 이하인 사람을 UNION으로 출력

```
SELECT cust_id, contact_name, mileage, city FROM customer WHERE city = '부
산'
UNION
SELECT cust_id, contact_name, mileage, city FROM customer WHERE mileage <=
1000 ORDER BY 1;
```

- **설명**: `UNION`은 두 쿼리의 결과를 합칩니다. 두 조건에 따라 각각 검색된 결과가 하나의 결과로 출력됩니다. `ORDER BY 1`은 첫 번째 열 기준으로 정렬합니다.

13. 특정 고객의 지역을 NULL로 수정

```
UPDATE customer SET region = NULL WHERE cust_id = 4;
```

- **설명**: 고객 ID가 4인 고객의 지역(region) 값을 `NULL`로 수정합니다. `NULL`은 값이 없음을 나타냅니다.

14. 특정 고객의 지역을 빈 문자열로 수정

```
UPDATE customer SET region = '' WHERE cust_id = 13;
```

- **설명**: 고객 ID가 13인 고객의 지역을 빈 문자열로 수정합니다. 빈 문자열은 데이터가 있지만 내용이 없음을 의미합니다.

15. 지역 데이터가 없는 고객 출력

```
SELECT * FROM customer WHERE region IS NULL;
```

- **설명**: 지역(region) 데이터가 `NULL`인 고객을 검색합니다. `NULL`은 값이 없는 상태를 의미합니다.

16. 지역 데이터가 빈 문자열인 고객 출력

```
SELECT * FROM customer WHERE region = '';
```

- **설명**: 지역 데이터가 빈 문자열인 고객을 검색합니다. `''`는 값이 빈 상태를 의미합니다.

17. 지역이 빈 문자열인 고객의 지역을 NULL로 수정

```
UPDATE customer SET region = NULL WHERE region = '';
```

- **설명**: 지역이 빈 문자열인 고객의 지역을 `NULL`로 수정합니다.

18. 직위가 대표이거나 사원인 고객 출력 (OR 사용)

```
SELECT cust_id, contact_name, contact_position FROM customer WHERE
contact_position = '대표' OR contact_position = '사원';
```

- **설명**: `OR` 조건을 사용하여 직위가 '대표'이거나 '사원'인 고객을 출력합니다.

19. 직위가 대표이거나 사원인 고객 출력 (IN 사용)

```
SELECT cust_id, contact_name, contact_position FROM customer WHERE
contact_position IN ('사원', '대표');
```

- **설명**: `IN`을 사용하여 직위가 '사원' 또는 '대표'인 고객을 검색합니다. `IN`은 OR 조건의 대체로 사용됩니다.

20. 마일리지가 100에서 200 사이인 고객 검색

```
SELECT contact_name, mileage FROM customer WHERE mileage BETWEEN 100 AND
200;
```

- **설명**: `BETWEEN` 연산자를 사용하여 마일리지가 100 이상 200 이하인 고객을 검색합니다.

21. 지역이 광역시인 고객 검색

```
SELECT * FROM customer WHERE region LIKE '%광역시';
```

- **설명**: `LIKE` 연산자를 사용하여 '광역시'라는 단어가 포함된 지역을 가진 고객을 검색합니다. `%`는 와일드카드로, 어떤 문자열이든 포함될 수 있음을 의미합니다.

22. city가 서울이고 마일리지가 15000에서 20000 사이인 고객 검색

```
SELECT * FROM customer WHERE city LIKE '서울%' AND mileage BETWEEN 15000
AND 20000;
```

- **설명**: 서울로 시작하는 도시 이름을 가진 고객 중 마일리지가 15000에서 20000 사이인 고객을 검색합니다. `LIKE`는 문자열 매칭을 위한 연산자입니다.

23. 지역이 중복되지 않도록 출력

```
SELECT DISTINCT region, city FROM customer ORDER BY 1, 2;
```

- **설명**: 지역과 도시를 중복 없이 출력합니다. `ORDER BY 1, 2`는 첫 번째와 두 번째 열을 기준으로 정렬합니다.

24. city가 서울, 대전이고 직위가 대표 또는 사원인 고객 검색

```
SELECT * FROM customer WHERE city IN ('서울', '대전') AND (contact_position
LIKE '%대표%' OR contact_position LIKE '%사원%');
```

- **설명**: 도시가 '서울' 또는 '대전'이면서 직위가 '대표' 또는 '사원'인 고객을 검색합니다. `LIKE` 연산자는 문자열 패턴 매칭을 위해 사용되었습니다.

25. 지역에 특별시 또는 광역시가 포함된 고객을 마일리지 역순으로 출력

```
SELECT * FROM customer WHERE region LIKE '%광역시' OR region LIKE '%특별시'
ORDER BY mileage DESC LIMIT 3;
```

- **설명**: 지역이 ‘광역시’ 또는 ‘특별시’로