[문제 1] CREATE 명령어를 이용하여 문제에서 요구하는 바를 그대로 작성하여 해결할 수 있다.

```
CREATE TABLE registered_member (
  id INT PRIMARY KEY,
  name VARCHAR(20) NOT NULL,
  age INT NOT NULL,
  gender VARCHAR(10) NOT NULL,
  address VARCHAR(100),
  grade INT NOT NULL,
  datetime DATETIME NOT NULL
);

CREATE TABLE retired_member (
  id INT PRIMARY KEY,
  name VARCHAR(20) NOT NULL,
  age INT NOT NULL,
  gender VARCHAR(10) NOT NULL,
  address VARCHAR(100),
  grade INT NOT NULL,
  datetime DATETIME NOT NULL
);
```

[문제 2] INSERT 명령어를 이용해 문제에서 요구하는 바를 그대로 작성하여 해결할 수 있다.

```
INSERT INTO registered_member
  VALUES
    (1, '나동빈', 30, '남성', '경기', 5, '2012-05-03 13:00:00'),
    (2, '홍길동', 20, '남성', '경기', 4, '2012-06-12 15:00:00'),
    (3, '김영희', 32, '여성', '서울', 3, '2012-08-14 21:00:00'),
```

```
    (4, '이순신', 24, '남성', '서울', 5, '2012-07-14 14:00:00'),
    (5, '임꺽정', 45, '남성', '인천', 2, '2012-08-08 15:00:00'),
    (6, '임민정', 33, '여성', '인천', 5, '2012-07-29 11:00:00'),
    (7, '김성민', 37, '남성', NULL, 4, '2012-10-21 13:00:00'),
    (8, '박지은', 33, '여성', '서울', 3, '2012-05-08 14:00:00'),
    (9, '이선희', 32, '여성', NULL, 1, '2012-09-26 12:00:00');

INSERT INTO retired_member
  VALUES
    (1, '나동빈', 30, '남성', '경기', 4, '2012-08-03 13:00:00'),
    (2, '홍길동', 20, '남성', '경기', 5, '2012-02-12 15:00:00'),
    (3, '김영희', 32, '여성', '서울', 2, '2012-11-14 21:00:00'),
    (4, '이순신', 24, '남성', '서울', 4, '2012-08-14 14:00:00'),
    (5, '임꺽정', 45, '남성', '인천', 3, '2012-09-08 15:00:00'),
    (21, '강민철', 33, '여성', '경기', 2, '2012-12-07 12:00:00'),
    (22, '김성민', 39, '남성', '서울', 5, '2012-10-05 10:00:00');
```

[문제 3] SELECT 명령어의 ORDER BY 옵션을 사용하여 정렬된 결과를 출력할 수 있다.

```
SELECT * FROM registered_member ORDER BY name;
```

[문제 4] SELECT 명령어의 WHERE 및 ORDER BY 문법을 사용하여 결과를 출력할 수 있다.

```
SELECT name FROM registered_member WHERE age >= 30 ORDER BY name;
```

[문제 5] SELECT 명령어에서 LIMIT 옵션을 함께 사용하여 해결할 수 있다.

```
SELECT name, datetime FROM registered_member ORDER BY datetime DESC LIMIT 1;
```

[문제 6] SELECT 명령어에서 ORDER BY 문법을 사용하여 해결할 수 있다.

```
SELECT * FROM registered_member ORDER BY grade DESC, datetime ASC;
```

[문제 7] MIN() 함수를 이용하여 해결할 수 있다.

```
SELECT MIN(grade) FROM registered_member;
```

[문제 8] COUNT() 함수를 이용하여 해결할 수 있다.

```
SELECT COUNT(id) FROM registered_member;
```

[문제 9] COUNT() 함수에 DISTINCT 옵션을 넣어 해결할 수 있다.

```
SELECT COUNT(DISTINCT grade) FROM registered_member;
```

[문제 10] GROUP BY 문법을 이용하여 해결할 수 있다.

```
SELECT grade, COUNT(id)
FROM registered_member
GROUP BY grade
ORDER BY grade;
```

[문제 11] GROUP BY 문법을 이용하여 해결할 수 있다.

```
SELECT grade, COUNT(id)
FROM registered_member
GROUP BY grade
HAVING COUNT(id) >= 2
```

```
ORDER BY grade;
```

[문제 12] SELECT 명령어의 WHERE 문법을 사용해 해결할 수 있다.

```
SELECT *
FROM registered_member
WHERE datetime >= '2012-07-01'
  AND datetime <= '2012-09-30'
ORDER BY datetime;
```

[문제 13] GROUP BY 문법을 이용하여 해결할 수 있다.

```
SELECT month(datetime), count(id)
FROM registered_member
WHERE datetime >= '2012-07-01'
  AND datetime <= '2012-09-30'
GROUP BY month(datetime)
ORDER BY datetime;
```

[문제 14] 일단 1월부터 8월까지의 각 달을 추출하는 SELECT 문을 작성해 보자. 아래처럼 코드를 작성하면 month 변수의 값이 1부터 8까지 증가하며, 각 값이 결과에 담기게 된다.

```
SET @month = 0;

SELECT (@month := @month + 1) AS month
FROM registered_member
WHERE @month < 8;
```

이제 여기에서 SELECT 구문을 하나 더 추가할 수 있다. 각 month에 대하여 실행되는 하나의

내부 SELECT 구문을 사용하자. 이는 중첩 질의(nested SELECT)로 이해할 수 있다. 따라서 정답은 다음과 같다.

```
SET @month = 0;

SELECT (@month := @month + 1) AS month,
  (SELECT COUNT(id) FROM registered_member WHERE MONTH(datetime) = @month) AS count
FROM registered_member
WHERE @month < 8;
```

혹은 다른 풀이 방법도 있다. WITH RECURSIVE 문법을 사용하여, 1부터 8까지의 값이 담긴 결과를 얻어 사용할 수 있다.

```
WITH RECURSIVE temp AS (
  SELECT 1 AS month
  UNION ALL
  SELECT month + 1 FROM temp
  WHERE month < 8
)

SELECT month, COUNT(id) AS count
FROM temp
LEFT JOIN registered_member
  ON MONTH(datetime) = temp.month
GROUP BY month
ORDER BY month;
```

[문제 15] NULL을 사용하여 지역 정보가 없는 회원들의 정보를 조회할 수 있다.

```
SELECT id
FROM registered_member
WHERE address IS NULL
ORDER BY id;
```

[문제 16] NOT NULL을 사용하여 지역 정보가 있는 회원들의 정보를 조회할 수 있다.

```
SELECT id
FROM registered_member
WHERE address IS NOT NULL
ORDER BY id;
```

[문제 17] IFNULL() 함수를 이용하면, NULL 값을 다른 문자열로 치환할 수 있다.

```
SELECT
  name,
  IFNULL(address, "지역 정보 없음") AS address,
  grade
FROM registered_member
ORDER BY name;
```

[문제 18] SELECT 명령어의 WHERE 문법을 사용해 해결할 수 있다.

```
SELECT
  name,
  address,
  gender,
  datetime
FROM registered_member
```

```
WHERE address = '경기'
ORDER BY name;
```

[문제 19] CASE 조건문을 사용해 해결할 수 있다.

```
SELECT name,
  (
    CASE
      WHEN age < 30 THEN 'young'
      ELSE 'old'
    END
  ) AS age
FROM registered_member
ORDER BY name;
```

[문제 20] 기본적인 INNER JOIN을 사용하여 문제를 해결할 수 있다. 테이블 A와 테이블 B에 동일한 ID가 있는 레코드에 대해서만 조회하게 된다.

```
SELECT registered_member.id, registered_member.name
FROM registered_member, retired_member
WHERE registered_member.id = retired_member.id
  AND registered_member.datetime > retired_member.datetime
ORDER BY registered_member.id;
```

[문제 21] 기본적인 LEFT JOIN을 사용하여 문제를 해결할 수 있다. LEFT JOIN은 테이블 A의 모든 레코드를 조회하되, 테이블 B에 동일한 ID가 있는 레코드와 JOIN한다.

```
SELECT retired_member.id, retired_member.name
FROM retired_member
```

```
LEFT JOIN registered_member
  ON retired_member.id = registered_member.id
WHERE registered_member.id IS NULL
ORDER BY retired_member.id;
```

혹은 CROSS JOIN을 사용하면 더 짧은 코드로도 해결할 수 있다.

```
SELECT DISTINCT retired_member.id, retired_member.name
FROM retired_member, registered_member
WHERE retired_member.id NOT IN (SELECT id FROM registered_member)
ORDER BY retired_member.id;
```

[문제 22] 기본적인 LEFT JOIN을 사용하여 문제를 해결할 수 있다. LEFT JOIN은 테이블 A의 모든 레코드를 조회하되, 테이블 B에 동일한 ID가 있는 레코드와 JOIN한다.

```
SELECT registered_member.name, registered_member.age, registered_member.datetime
FROM registered_member
LEFT JOIN retired_member
  ON registered_member.id = retired_member.id
WHERE retired_member.id IS NULL
ORDER BY registered_member.datetime
LIMIT 2;
```

[문제 23] 기본적인 INNER JOIN을 사용하여 문제를 해결할 수 있다. 테이블 A와 테이블 B에 동일한 ID가 있는 레코드에 대해서만 조회하게 된다.

```
SELECT retired_member.name, retired_member.datetime, retired_member.grade
FROM registered_member, retired_member
WHERE registered_member.id = retired_member.id
```

```
  AND registered_member.grade > retired_member.grade
ORDER BY registered_member.name;
```

[문제 24] GROUP BY 문법을 이용하여 해결할 수 있다.

```
SELECT gender, AVG(age), AVG(grade)
FROM registered_member
GROUP BY gender
ORDER BY AVG(grade);
```

[문제 25] LIKE 문법을 이용하여 해결할 수 있다.

```
SELECT name, grade, age, datetime
FROM registered_member
WHERE name LIKE "%동빈"
  OR name LIKE "%길동"
  OR name LIKE "%민정"
  OR name LIKE "%성민"
ORDER BY datetime;
```

[문제 26] IN 문법을 이용하여 해결할 수 있다.

```
SELECT name, grade, age, datetime
FROM registered_member
WHERE name IN ('이순신', '홍길동', '김성민')
ORDER BY datetime;
```

[문제 27] LIKE 문법을 이용하여 해결할 수 있다.

```
SELECT name, grade, age, datetime
FROM retired_member
WHERE name LIKE '%민%'
  AND gender = "남성"
ORDER BY datetime;
```

[문제 28] WHERE 문법을 사용하여 해결할 수 있다.

```
SELECT registered_member.name, registered_member.grade
FROM registered_member, retired_member
WHERE registered_member.id = retired_member.id
ORDER BY
  (retired_member.datetime - registered_member.datetime) DESC,
  name ASC
LIMIT 3;
```

[문제 29] DATE_FORMAT 함수를 사용하면 DATETIME 형식을 자유롭게 변경할 수 있다.

```
SELECT name, DATE_FORMAT(datetime, '%Y-%m-%d') as date
FROM registered_member
WHERE gender = '남성'
ORDER BY datetime;
```