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

```
# DROP DATABASE my_database;

# CREATE DATABASE my_database CHARACTER SET UTF8 COLLATE UTF8_general_ci;

# USE my_database;


CREATE TABLE employee (

  employee_id INT PRIMARY KEY,

  name VARCHAR(30) NOT NULL,

  age INT NOT NULL,

  gender VARCHAR(30) NOT NULL,

  address VARCHAR(100) NOT NULL

);


CREATE TABLE company (

  company_id INT PRIMARY KEY,

  name VARCHAR(30) NOT NULL,

  category VARCHAR(30) NOT NULL,

  address VARCHAR(100) NOT NULL

);


CREATE TABLE affiliation (

  affiliation_id INT PRIMARY KEY,

  employee_id INT NOT NULL,

  company_id INT NOT NULL,

  pay INT NOT NULL,

  entry_date DATE NOT NULL,

  FOREIGN KEY (employee_id) REFERENCES employee (employee_id),

  FOREIGN KEY (company_id) REFERENCES company (company_id)
);
```

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

```
INSERT INTO employee
  VALUES
    (1, '나동빈', 27, '남성', '경기'),
    (2, '홍길동', 20, '남성', '경기'),
    (3, '김영희', 32, '여성', '서울'),
    (4, '이순신', 24, '남성', '서울'),
    (5, '임꺽정', 45, '남성', '인천'),
    (6, '임민정', 33, '여성', '인천'),
    (7, '김성민', 37, '남성', '경기'),
    (8, '박지은', 33, '여성', '서울'),
    (9, '이선희', 32, '여성', '경기');

INSERT INTO company
  VALUES
    (1, '천국테크', 'IT', '서울'),
    (2, '내일은개발왕', 'IT', '경기'),
    (3, '서울방송', '방송', '서울'),
    (4, 'K디자인', '디자인', '인천'),
    (5, '빛나리', '디자인', '서울');

INSERT INTO affiliation
  VALUES
    (1, 1, 1, 3000, '2012-05-09'),
    (2, 2, 1, 5000, '2012-05-21'),
    (3, 3, 1, 4500, '2012-08-11'),
    (4, 4, 2, 6500, '2012-05-14'),
    (5, 5, 2, 7000, '2012-04-23'),
    (6, 6, 3, 4000, '2012-09-15'),
    (7, 7, 4, 3500, '2012-05-06'),
    (8, 8, 4, 5500, '2012-08-08'),
    (9, 9, 5, 4500, '2012-08-07');
```

[문제 3] JOIN 구문을 사용하여 문제를 해결할 수 있다.

```
SELECT e.name, e.address
  FROM employee as e, company as c, affiliation as a
  WHERE e.employee_id = a.employee_id
    AND c.company_id = a.company_id
    AND c.name = '천국테크'
  ORDER BY e.name;
```

[문제 4] JOIN 구문을 사용하여 문제를 해결할 수 있다.

```
SELECT e.name, c.name, c.address, e.address
  FROM employee AS e, company AS c, affiliation AS a
  WHERE e.employee_id = a.employee_id
    AND c.company_id = a.company_id
    AND e.address != c.address
  ORDER BY e.name;
```

[문제 5] JOIN 구문을 사용하여 문제를 해결할 수 있다.

```
SELECT c.name, COUNT(e.employee_id), AVG(a.pay)
  FROM employee AS e, affiliation AS a, company AS c
  WHERE e.employee_id = a.employee_id
    AND c.company_id = a.company_id
  GROUP BY c.company_id
  ORDER BY c.name;
```

[문제 6] 기본적으로 각 회사의 평균 급여는 다음과 같이 알 수 있다.

```
SELECT temp.company_id, AVG(temp.pay) AS pay
  FROM affiliation AS temp
  GROUP BY temp.company_id;
```

따라서 이것을 내장 SELECT 구문으로 사용하면 다음과 같이 정답 코드를 작성할 수 있다.

```
SELECT e.name, c.company_id, a.pay
  FROM employee AS e, affiliation AS a,
    (
      SELECT temp.company_id as company_id, AVG(temp.pay) AS avg_pay
        FROM affiliation AS temp
        GROUP BY temp.company_id
    ) AS c
  WHERE e.employee_id = a.employee_id
    AND c.company_id = a.company_id
    AND a.pay > c.avg_pay
  ORDER BY e.name;
```

[문제 7] 'K디자인' 회사에서 가장 낮은 급여액보다 높은 급여를 받는 모든 사람을 모두 출력하는 것과 같다. 이는 ANY 구문을 사용하여 해결할 수 있다.

```
SELECT e.name, c.name, a.pay
```

```sql
FROM employee AS e, company as c, affiliation as a
WHERE e.employee_id = a.employee_id
  AND c.company_id = a.company_id
  AND a.pay > ANY (
    SELECT temp2.pay
      FROM company as temp1, affiliation as temp2
      WHERE temp1.company_id = temp2.company_id
        AND temp1.name = 'K디자인'
  )
ORDER BY a.pay;
```