ECE-568-Homework-1

Name: Zekun Zhang

NetID: zz364

Email: zekunzhang.1996@gmail.com

Question 1

(1)

ERD for Homework-1-Question-1

(2)

```
CREATE TABLE employee
(
    SSN            CHAR(11) NOT NULL,
    name           CHAR(20),
    specialization CHAR(20),
    PRIMARY KEY (SSN)
);

CREATE TABLE gym
(
    name        CHAR(20) NOT NULL,
    street_no   INT,
    street_name CHAR(40),
    zip_code    INT,
    manager     CHAR(11) NOT NULL,
    FOREIGN KEY (manager) REFERENCES employee (SSN),
    PRIMARY KEY (name)
);
```

```
CREATE TABLE customer
(
    SSN  CHAR(11) NOT NULL,
    name CHAR(20),
    age  INT,
    PRIMARY KEY (SSN)
);


CREATE TABLE phone_no
(
    phone_no INT NOT NULL,
    gym      CHAR(20),
    FOREIGN KEY (gym) REFERENCES gym (name),
    PRIMARY KEY (name)
);

CREATE TABLE certifications
(
    employee           CHAR(11) NOT NULL,
    certification_name CHAR(20) NOT NULL,
    PRIMARY KEY (employee, certification_name),
    FOREIGN KEY (employee) REFERENCES employee (SSN)
);

CREATE TABLE go_to
(
    customer CHAR(11),
    gym      CHAR(20),
    PRIMARY KEY (customer, gym),
    FOREIGN KEY (customer) REFERENCES customer (SSN),
    FOREIGN KEY (gym) REFERENCES gym (name)
);

CREATE TABLE guest
(
    name     CHAR(20),
    age      INT,
    customer CHAR(11),
    PRIMARY KEY (customer, name, age),
    FOREIGN KEY (customer) REFERENCES customer (SSN)
);

CREATE TABLE work
(
    gym         CHAR(20) NOT NULL,
    employee    CHAR(11) NOT NULL,
    percetage   REAL NOT NULL,
    PRIMARY KEY (gym, employee, percetage),
    FOREIGN KEY (gym) REFERENCES gym (name),
    FOREIGN KEY (employee) REFERENCES employee (SSN)
);
```

Question 2

(1)

```
SELECT s.sname
FROM suppliers s
WHERE NOT EXISTS (SELECT p.pid
                    FROM parts p
                    WHERE p.pid NOT IN
                        (SELECT c.pid
                            FROM catalog c
                            WHERE c.sid = s.sid));
```

(2)

```
SELECT c.sid
FROM catalog c
WHERE c.cost > (SELECT AVG(c1.cost)
                FROM catalog c1
                WHERE c1.pid = c.pid);
```

(3)

```
SELECT s.sname
FROM suppliers s
WHERE EXISTS (SELECT c.sid
                FROM catalog c
                WHERE c.cost IN
                (SELECT MAX(c1.cost)
                    FROM (SELECT c1.cost,
                            FROM catalog c1
                            WHERE c1.pid = c.pid)));
```

(4)

```sql
SELECT c.sid
FROM catalog c
WHERE NOT EXISTS (SELECT p.color
                    FROM parts p
                    WHERE p.color <> "red"
                    AND p.pid = c.pid);
```

(5)

```sql
SELECT c.sid
FROM catalog c
WHERE EXISTS (SELECT p.color
                FROM parts p
                WHERE p.color = "green"
                OR p.color = "red"
                AND p.pid = c.pid);
```

(6)

```sql
SELECT s.sname,MAX(c.cost)
FROM suppliers s, parts p, catalog c
WHERE p.color = "red"
    AND p.color = "green"
    AND p.pid = c.pid
    AND s.sid = c.sid;
```

Question 3

(1)

```
SELECT m.moviename
FROM movies m, moviesupplier ms, suppliers s
WHERE m.movieid = ms.movieid
    AND ms.supplierid = s.supplierid
    AND (suppliername = "Ben's Video"
    OR suppliername = "Video Clubhouse");
```

(2)

```
SELECT m.moviename
FROM movies m, inventory i, rentals r
WHERE m.movieid = i.movieid
    AND i.tapeid = r.tapeid
    AND r.duration >= ALL(SELECT duration
                        FROM rentals));
```

(3)

```
SELECT s.suppliername
FROM suppliers s
WHERE s.supplierid IN
    (SELECT ms.supplierid
        FROM moviesupplier ms
        WHERE NOT EXISTS (SELECT i.movieid
                        FROM inventory i
                        WHERE i.movieid NOT IN
                        (SELECT ms1.movieid
                        FROM moviesupplier ms1
                        INNER JOIN inventory i1
                        ON i1.movieid = ms1.movieid)));
```

(4)

```
SELECT s.suppliername, COUNT(DISTINCT movieid)
FROM suppliers s, moviesupplier ms, inventory i
WHERE i.movieid = ms.movieid
    AND s.supplierid = ms.suppilerid;
```

(5)

```
SELECT m.moviename
FROM movie m, orders o
WHERE m.movieid = o.movieid
GROUP BY m.moviename
HAVING SUM(o.copies) > 4;
```

(6)

```
SELECT c.lastname, c.firstname
FROM customers c, rentals r, inventory i, movies m
WHERE c.customerid = r.customerid
    AND r.tapeid = i.tapeid
    AND i.movieid = m.movieid
    AND m.moviename = "Kung Fu Panda"
UNION
SELECT c.lastname, c.firstname
FROM customers c, rentals r, inventory i, moviesupplier ms, suppliers s
WHERE c.customerid = r.customerid
    AND r.tapeid = i.tapeid
    AND i.movieid = ms.movieid
    AND ms.supplierid = s.supplierid
    AND s.suppliername = "Palm Video";
```

(7)

```
SELECT m.moviename
FROM movies m, inventory i1, inventory i2
WHERE m.movieid = i1.movieid
    AND i1.movieid = i2.movieid
    AND i1.tapeid <> i2.tapeid;
```

(8)

```
SELECT c.lastname, c.firstname
FROM customers c, rentals r
WHERE r.duration >= 5
    AND c.customerid = r.customerid;
```

(9)

```
SELECT s.suppliername
FROM suppliers s, moviesupplier ms, movie m
WHERE ms.price <= ALL(SELECT price
                        FROM moviesupplier ms1, movie m1
                        WHERE m1.moviename = "Cinderella 2015"
                            AND m1.movieid = ms1.movieid)
    AND s.supplierid = ms.supplierid
    AND ms.movieid = m.movieid
    AND m.moviename = "Cinderella 2015";
```

(10)

```
SELECT m.moviename
FROM movies m
WHERE m.movieid NOT IN
    (SELECT i.movieid
        FROM inventory i);
```

Question 4

(a)

In this question, we can easily find that TRIGGER is actually doing some recursively work. According to this feature, we can use these codes to describe what happens in these cases.

```
UPDATE purchase
SET price = 1.5
WHERE purchaseID = 111

UPDATE purchase
SET price = 3
WHERE purchaseID = 111
```

(b)

```
UPDATE purchase
SET price = 3
WHERE purchaseID = 111

UPDATE purchase
SET price = 1.5
WHERE purchaseID = 111
```

(c)

```
UPDATE purchase
SET price = 1.5
WHERE purchaseID = 111
```