# HW1

## Q1

### (1)

ER Diagram is as follow:



No overlap constraint, for every *employee* has at most one specialization.

No covering constraint, for some *employees* may have zero specialization.

### (2)

The SQL statements are as follow:

CREATE TABLE gym(  
 name char(20),  
 strNum integer,  
 strName char(20)  
 ZIP integer,  
 PRIMARY KEY (name)  
);  
CREATE TABLE phoneNum(  
 phoneNum long integer,  
 gymName char(20),  
 PRIMARY KEY (phoneNum),  
 FOREIGN KEY (gymName) REFERENCES gym  
);  
CREATE TABLE employee(  
 SSN char(11),  
 name char(20),  
 specialization char(20),  
 PRIMARY KEY (SSN)  
);  
CREATE TABLE workIn(  
 gymName char(20),  
 employeeSSN char(11),  
 timePts integer,  
 PRIMARY KEY (gymName,employeeSSN),  
 FOREIGN KEY (gymName) REFERENCES gym,  
 FOREIGN KEY (employeeSSN) REFERENCE employee  
);  
CREATE TABLE manages(  
 gymName char(20),  
 employeeSSN char(11),  
 PRIMARY KEY (gymName),  
 FOREIGN KEY (gymName) REFERENCES gym,  
 FOREIGN KEY (employeeSSN) REFERENCE employee  
);  
CREATE TABLE customer(  
 SSN char(11),  
 name char(20),  
 age integer,  
 PRIMARY KEY (SSN)  
);  
CREATE TABLE goesTo(  
 gymName char(20),  
 customerSSN char(11),  
 PRIMARY KEY (gymName,customerSSN),  
 FOREIGN KEY (gymName) REFERENCES gym,  
 FOREIGN KEY (customerSSN) REFERENCE customer  
);  
CREATE TABLE guest(  
 name char(20),  
 age integer,  
 hostSSN char(11),  
 PRIMARY KEY (name,age),  
 FOREIGN KEY (hostSSN) REFERENCES customer  
);

## Q2

### (1)

SELECT s.sname  
FROM Suppliers s, Catalog c  
WHERE s.sid=c.sid  
GROUP BY s.sid   
HAVING COUNT(c.pid) = (SELECT COUNT(\*) FROM parts);

### (2)

SELECT DISTINCT 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, Parts P, Catalog C  
WHERE C.sid = S.sid AND C.pid = P.pid  
 AND C.cost = (SELECT MAX(C1.cost)  
 FROM Catalog C1  
 WHERE C1.pid = C.pid);

### (4)

SELECT DISTINCT C.sid  
FROM Catalog C, Parts P  
WHERE C.pid = P.pid AND P.color = "red";

### (5)

SELECT DISTINCT C.sid  
FROM Catalog C, Parts P  
WHERE C.pid = P.pid AND (P.color = "red" OR P.color = "green");

### (6)

SELECT S.sname, MAX(C.cost)  
FROM Suppliers S, Catalog C, Parts P  
WHERE S.sid = C.sid AND C.pid = P.pid   
 AND P.color IN ("red","green");

## Q3

### (1)

SELECT MS.MovieID  
FROM MovieSupplier MS, Suppliers S  
WHERE MS.SupplierID = S.SupplierID  
 AND S.SupplierName IN ("Ben's Video","Video Clubhouse");

### (2)

SELECT I.MovieID  
FROM Inventory I, Rentals R  
WHERE I.TapeID = R.TapeID AND R.Duration = (SELECT MAX(Duration)  
 FROM Rentals);

### (3)

SELECT MS.SupplierID  
FROM Inventory I LEFT JOIN MovieSupplier MS ON I.MovieID=MS.MovieID  
GROUP BY MS.SupplierID  
HAVING COUNT(MS.MovieID) = (SELECT COUNT(DISTINCT MovieID)  
 FROM Inventory);

### (4)

SELECT MS.SupplierID, COUNT(I.MovieID)  
FROM MovieSupplier MS, Inventory I  
WHERE MS.MovieID = I.MovieID  
GROUP BY MS.SupplierID;

### (5)

SELECT O.MovieID  
FROM Orders O  
WHERE SUM(O.Copies) > 4  
GROUP BY O.MovieID;

### (6)

SELECT R.CustomerID  
FROM Rentals R, Inventory I, Movies M  
WHERE R.TapeID = I.TapeID AND I.MovieID = M.MovieID  
 AND M.MovieName = "Kung Fu Panda"  
UNION  
SELECT R.CustomerID  
FROM Rentals R, Inventory I, MovieSupplier MS, Suppliers S  
WHERE R.TapeID = I.TapeID AND I.MovieID = MS.MovieID  
 AND MS.SupplierID = S.SupplierID  
 AND S.SupplierName = "Palm Video";

### (7)

SELECT I.MovieID  
FROM Inventory I  
WHERE COUNT(I.TapeID) > 1  
GROUP BY I.MovieID;

### (8)

SELECT R.CustomerID  
FROM Rentals R  
WHERE R.Duration >= 5;

### (9)

SELECT MS.SupplierID  
FROM MovieSupplier MS, Movies M  
WHERE MS.MovieID = M.MovieID AND M.MovieName = "Cinderella 2015"  
 AND MS.Price = MIN(SELECT Price  
 FROM MovieSupplier  
 WHERE MovieID = M.MovieID);

### (10)

SELECT MovieID  
FROM Movies  
EXCEPT  
SELECT MovieID  
FROM Inventory;

## Q4

### (1)

First, it would set (111,4) to (111,1.5), because 4>3>1 which meets the trigger requirement:

(OldTuple.price > NewTuple.price AND NewTuple.price > 1)

And then, it would complete the original update statement to set (111,1.5) to (111,3), because the trigger is run *BEFORE* the update

### (2)

First, (111,4) --> (111,3)

Then, (111,3) --> (111,1.5)

### (3)

It would only do (111,4) --> (111,1.5)