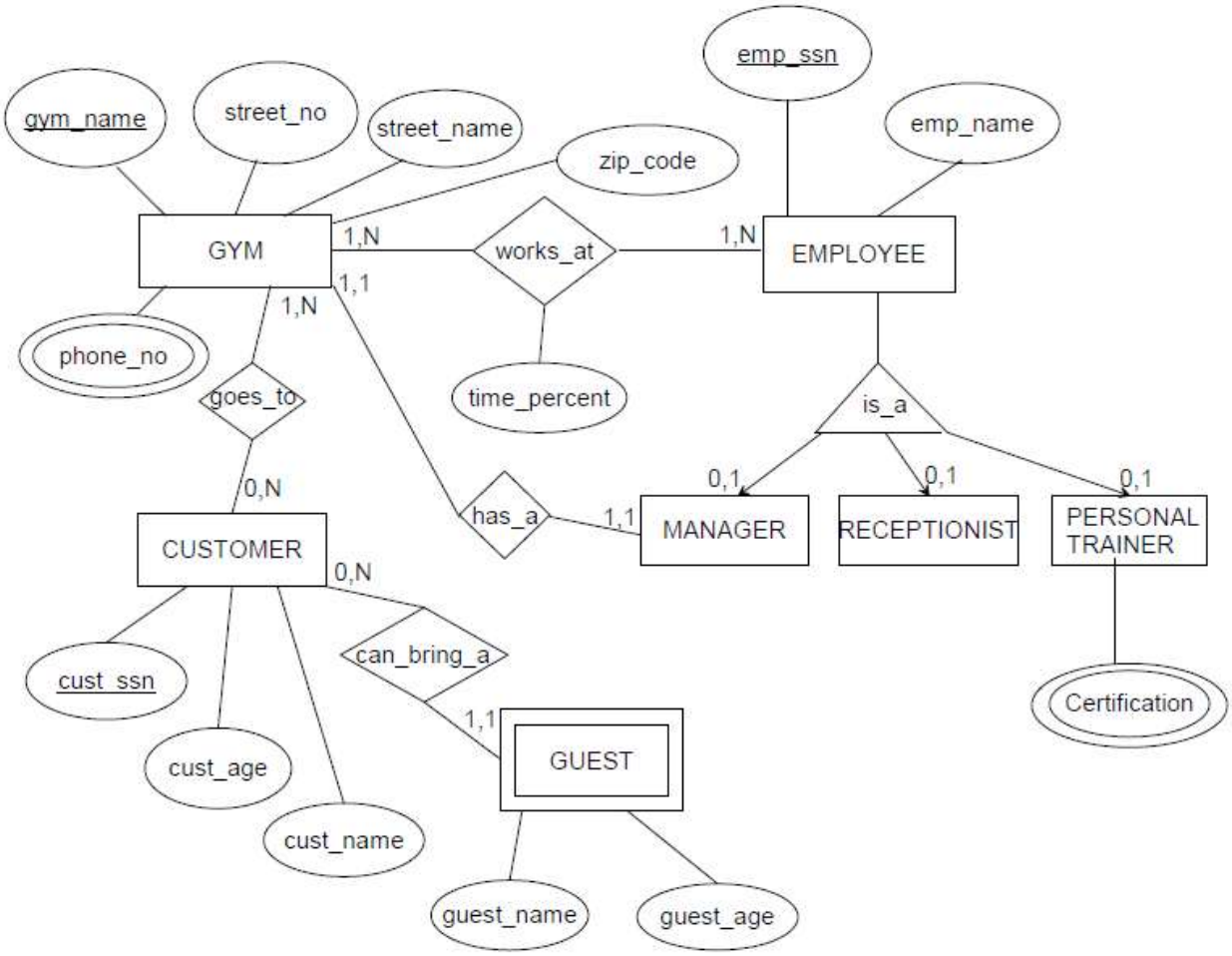
SOFTWARE ENGINEERING FOR WEB APPLICATIONS

HOMEWORK-1

1. a. The ER diagram for the gym database is shown below:



b. The conversion from the ER diagram to tables is as follows:

CREATE TABLE GYM (

gym_name **VARCHAR(20) NOT NULL PRIMARY KEY,**

street_no **INT,**

street_name **VARCHAR(20),**

zip_code **INT**,

)

CREATE TABLE EMPLOYEE (

emp_ssn **VARCHAR(20) NOT NULL PRIMARY KEY**,

emp_name **VARCHAR(20)**

)

CREATE TABLE Works_at (

emp_ssn **VARCHAR(20) NOT NULL**,

gym_name **VARCHAR(20) NOT NULL**,

time_percent **FLOAT NULL**,

PRIMARY KEY (emp_ssn_gym_name)

FOREIGN KEY emp_ssn **REFERENCES** EMPLOYEE (emp_ssn),

FOREIGN KEY gym_name **REFERENCES** GYM (gym_name)

)

CREATE TABLE MANAGER (

emp_ssn **VARCHAR(20) NOT NULL PRIMARY KEY**,

FOREIGN KEY emp_ssn **REFERENCES** EMPLOYEE (emp_ssn)

)

CREATE TABLE RECEPTIONIST (

emp_ssn **VARCHAR(20) NOT NULL PRIMARY KEY**,

FOREIGN KEY emp_ssn **REFERENCES EMPLOYEE** (emp_ssn)

)

CREATE TABLE PERSONAL TRAINER (

```
emp_ssn VARCHAR(20) NOT NULL PRIMARY KEY,
Certification VARCHAR(20) NULL,
FOREIGN KEY emp_ssn REFERENCES EMPLOYEE (emp_ssn)
)
CREATE TABLE CUSTOMER (
cust_ssn VARCHAR(20) NOT NULL PRIMARY KEY,
cust_age INT,
cust_name VARCHAR(20) NOT NULL
)

CREATE TABLE GUEST (
guest_age INT NOT NULL,
guest_name VARCHAR(20)
)

CREATE TABLE phone_no (
gym_name VARCHAR(20) NOT NULL,
phone_no VARCHAR(20) NOT NULL,
PRIMARY KEY (phone_no, gym_name),
FOREIGN KEY gym_name REFERENCES GYM (gym_name)
)

CREATE TABLE goes_to (
gym_name VARCHAR(20) NOT NULL,
cust_ssn VARCHAR(20) NOT NULL,
PRIMARY KEY (gym_name, cust_ssn)
FOREIGN KEY gym_name REFERENCES GYM (gym_name),
```

FOREIGN KEY cust_ssn **REFERENCES** CUSTOMER (cust_ssn)

)

2.

1. **SELECT** sname **FROM** Suppliers
   WHERE EXISTS(
   SELECT Catalog.sid from Catalog **WHERE** Catalog.sid=Suppliers.sid
   );

2. SELECT DISTINCT sid FROM Catalog C1
   WHERE C1.cost>(
   SELECT AVG(C2.cost) FROM Catalog C2
   WHERE C1.pid=C2.pid
   );

3. SELECT P.pid, 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
   WHERE NOT EXISTS(
   SELECT * FROM Parts P
   WHERE P.pid=C.pid
   AND P.color<>'red'
)

5. SELECT DISTINCT C.sid
   From Catalog C,Parts P
   WHERE C.pid=P.pid
   AND P.color='red'
   UNION
   SELECT DISTINCT C.sid
   FROM Catalog C1, Parts P1

```
WHERE  C1.pid=P1.pid
AND P1.color='green'
```

6.
```
SELECT P.pname ,MAX(C.cost) AS MAXCost FROM Suppliers S,Catalog C,Parts P
WHERE P.pid=C.pid
AND C.sid=S.sid
AND P.color IN ('red','green');
```

3.

1.
```
SELECT M.MovieName FROM MovieSupplier MS, Movies M,Suppliers S
WHERE M.MovieID=MS.MovieID
AND S.SupplierID=MS.SupplierID
AND S.SupplierName IN('Ben''s Video','Video Clubhouse');
```

2.
```
SELECT M.MovieName FROM Movies M,Rentals R,Inventory I
WHERE I.TapeIID=R.TapeIID
AND I.MovieID=M.MovieID
AND R.Duration >= ALL(SELECT Duration FROM Rentals);
```

3.
```
SELECT S.SupplierName FROM Suppliers S
WHERE S.SupplierID NOT IN (
 SELECT MS.SupplierID FROM MovieSupplier MS,Inventory I
 WHERE NOT EXISTS(
  SELECT * FROM Inventory I1,MovieSupplier MS1
  WHERE MS1.MovieID=I1.MovieID
  AND MS.SupplierID=MS1.SupplierID
  AND I1.MovieID=I.MovieID
 )
);
```

4.
```
SELECT S.SupplierName, COUNT(DISTINCT MovieID) FROM Suppliers S, Movies M,
MovieSupplier MS
where S.SupplierID=MS.SupplierID
AND M.MovieID=MS.MovieID
GROUP BY  S.SupplierName
```

5. SELECT M.MovieName FROM Movies M,Orders O
   WHERE O.MovieID=M.MovieID
   GROUP BY M.MovieName
   HAVING SUM(Copies)>4;

6. SELECT C.LastName,C.FirstnAME FROM Rentals R,Movies M,Inventory I, Customers C
   WHERE R.TapeIID=I.TapeIID
   AND C.CustID=R.CustomerID
   AND I.MovieID=M.MovieID
   AND MovieName='Kung Fu Panda'
   UNION
   SELECT C.LastName,C.FirstnAME
   FROM Rentals R1,Movies M1,Inventory I1, Customers C1,MovieSupplier MS1,Suppliers S1
   WHERE R1.TapeIID=I1.TapeIID
   AND C1.CustID=R1.CustomerID
   AND I1.MovieID=M1.MovieID
   AND MS1.MovieID=M1.MovieID
   AND MS1.SupplierID=S1.SupplierID
   AND S1.SupplierName='Palm Video'
7. SELECT M.MovieName
   FROM Inventory I1,Inventory I2,Movies M
   WHERE I1.MovieID=M.MovieID
   AND I1.MovieID=I2.MovieID
   AND I1.TapeIID<>I2.TapeIID;

8. SELECT C.FirstnAME,C.LastName
   FROM Customers C,Rentals R
   WHERE C.CustID=R.CustomerID
   AND Duration>=5;

9. SELECT S.SupplierName
   FROM Suppliers S, Movies M, MovieSupplier MS
   WHERE MS.MovieID=M.MovieID
   AND MS.SupplierID=S.SupplierID
   AND M.MovieName='Cinderella 2015'
   AND MS.Price<=ALL(
    SELECT MovieSupplier.Price
    FROM MovieSupplier , Movies
    WHERE MovieSupplier.MovieID=Movies.MovieID

```
  AND Movies.MovieName='Cinderella 2015'
);
```

10.
```
SELECT M.MovieName
FROM Movies
WHERE MovieID NOT IN (
  SELECT MovieID
  FROM Inventory
```

4.

a) On changing the tuple to (111, 3) from (111, 4), the trigger is completed and since OldTuple.price (4) is greater than NewTuple.price (3), the records are updated. The result becomes (111, 1.5).

b) If BEFORE is replaced by AFTER, the records in the table Purchases are first updated and then the condition is checked. The result is updated to (111, 1.5).

c) If BEFORE is replaced by INSTEAD OF, the condition is first tested and then the records in the table are updated in the table. The result is (111, 1.5).