WA2 Solutions

Problem 1

1.
```
SELECT      DISTINCT S.sname
FROM  Student S, Class C, Enrolled E, Faculty F
WHERE       S.snum = E.snum AND E.cname = C.cname AND C.fid = F.fid AND F.fname =
      'Jonathan' AND S.year = 'JR'
```

2.
```
SELECT      C.cname
FROM  Class C
WHERE       C.room = 'R128' OR
      C.cname IN (  SELECT            E.cname
                    FROM        Enrolled E
                    GROUP BY    E.cname
                    HAVING      COUNT (*) >= 5)
```

3.
```
SELECT      DISTINCT F.fname
FROM  Faculty F
WHERE       5 > (   SELECT      COUNT(E.snum)
                FROM  Enrolled E, Class C
                WHERE       C.cname = E.cname AND C.fid = F.fid)
```

4
```
SELECT      DISTINCT S.sname
FROM  Student S
WHERE       S.snum IN (   SELECT            E.snum
                    FROM        Enrolled E
                    GROUP BY    E.snum
                    HAVING      COUNT (*) >= ALL (  SELECT          COUNT(*)
                                                    FROM        Enrolled E1
                                                    GROUP BY    E1.snum))
```
5.
```
SELECT      DISTINCT S.sname
FROM  Student S
WHERE       S.snum NOT IN (     SELECT E.snum
                          FROM  Enrolled E)
```

Problem 2

There are many different ways to write each query.

1.
```
SELECT    DISTINCT MS.sname
FROM      MovieStar MS, Movie M, StarsIn SI
WHERE   MS.SNo = SI.SNo AND M.MNo = SI.MNo AND M.profit > 100M
```

2.
```
SELECT    DISTINCT MS.sname
FROM      MovieStar MS, Movie M1, Movie M2, StarsIn SI1, StarsIn SI2
WHERE MS.SNo = SI1.SNo AND MS.SNo = SI2.SNo AND SI1.MNo <> SI2.MNo AND
          M1.MNo = SI1.MNo AND M1.profit > 100M AND M2.MNo = SI2.MNo AND
          M2.profit > 100M
```

3.
```
SELECT MS.sname
FROM              MovieStar MS, Director DR
WHERE MS.sname = DR.dname
(or MS.SNo = DR.DNo)
```

4.
```
SELECT MS.sname, AVG(M.profit) AS AvgProfit
FROM MovieStar MS, Movie M, StarsIn SI WHERE
MS.SNo = SI.SNo AND SI.MNo = M.MNo GROUP BY
MS.SNo, MS.sname HAVING SUM(M.profit) > 2M
```

5.
```
SELECT MS.sname
FROM              MovieStar MS
WHERE NOT EXISTS ( ( SELECT DISTINCT M.genre
                                          FROM Movie M)
                                        EXCEPT
                                          ( SELECT M2.genre
                                           FROM StarsIn SI, Movie M2
                                               WHERE SI.SNo = MS.SNo AND SI.MNo = M2.MNo))
```

Problem 3.

There are many ways to create the constraints. Here we give some examples.

1. This constraint can be added by modifying the Emp table:

```
CREATE TABLE Emp     (   eid   INTEGER,
                         ename CHAR(20),
                         age INTEGER,
                         salary REAL,
                         PRIMARY KEY (eid),
                         CHECK (salary>1000))
```

2. Create an assertion as follows:

```
CREATE ASSERTION ManagerIsEmployee
CHECK ( ( SELECT COUNT(*)
          FROM Dept D
          WHERE D.managerid NOT IN ( SELECT eid FROM Emp))=0)
```

Another option is to create a constraint on the Dept table.

3. This constraint can be added by modifying the Works table:

```
CREATE TABLE Works ( eid INTEGER,
                     did INTEGER,
                     pcttime INTEGER,
                     PRIMARY KEY (eid, did),
                     CHECK ( NOT EXISTS( SELECT W.eid
                                         FROM Works W
                                         GROUP BY W.eid
                                         HAVING Sum(pcttime)>100))
```

Problem 4.

AB, BCH, BCF are the candidate keys

BCNF
R = {A,B,C,D,E,F,G,H}

ABCDEFGH
ABCEFGH, AD (from A->D)
ABCEFH, AD, FG (from F->G)
ABCEF, AD, FG, BFH (from BF->H)

BCNF = {ABCEF, AD, FG, BFH}

Lossless, Yes.

Not dependency preserving, because we cannot check: BCH-> A using a single table.

3NF

R = {A,B,C,D,E,F,G,H}

ABCDEFGH
ABCEFGH, AD (from A->D)
ABCEFH, AD, FG (from F->G)

Now, all relations are in 3NF!!!. AD, and FG are actually in BCNF and therefore also in 3NF. Relation ABCEFH is in 3NF, because the only FD where the left hand side has not a superkey is: BF->H. However H is part of a candidate key, the BCH. Therefore, this also satisfies the requirement for 3NF.

Another approach that I will consider correct is to do a BCNF decomposition and then add the extra table BCHA, so all the FDs can be checked with a single table. This decomposition is 3NF and DP (is actually BCNF and DP, but is not minimal.)

Here is also a minimal cover that you can use for 3NF:

{AB->F, A->D, F->G, BF->H, BCH->F, BCF->AE}

Notice that minimal cover is not unique.