Answer 5.7 The answers are given below:

1. Define a table constraint on Emp that will ensure that every employee makes at least $10,000

```
CREATE TABLE Emp ( eid      INTEGER,
                   ename    CHAR(10),
                   age      INTEGER ,
                   salary   REAL,
                   PRIMARY KEY (eid),
                   CHECK ( salary >= 10000 ))
```

2. Define a table constraint on Dept that will ensure that all managers have $age > 30$

```
CREATE TABLE Dept ( did         INTEGER,
                    buget       REAL,
                    managerid   INTEGER ,
                    PRIMARY KEY (did),
                    FOREIGN KEY (managerid) REFERENCES Emp,
                    CHECK (     ( SELECT E.age FROM Emp E, Dept D)
                                WHERE E.eid = D.managerid ) > 30 )
```

3. Define an assertion on Dept that will ensure that all managers have age > 30

```
CREATE TABLE Dept ( did         INTEGER,
                    budget      REAL,
                    managerid   INTEGER ,
                    PRIMARY KEY (did) )
```

```
CREATE ASSERTION managerAge
CHECK ((SELECT E.age
        FROM   Emp E, Dept D
        WHERE  E.eid = D.managerid ) > 30 )
```

Since the constraint involves two relations, it is better to define it as an assertion, independent of any one relation, rather than as a check condition on the Dept relation. The limitation of the latter approach is that the condition is checked only when the Dept relation is being updated. However, since age is an attribute of the Emp relation, it is possible to update the age of a manager which violates the constraint. So the former approach is better since it checks for potential violation of the assertion whenever one of the relations is updated.

4. To write such statements, it is necessary to consider the constraints defined over the tables. We will assume the following:

```
CREATE TABLE Emp (    eid          INTEGER,
                      ename        CHAR(10),
                      age          INTEGER,
                      salary       REAL,
                      PRIMARY KEY (eid) )

CREATE TABLE Works (  eid          INTEGER,
                      did          INTEGER,
                      pcttime      INTEGER,
                      PRIMARY KEY (eid, did),
                      FOREIGN KEY (did) REFERENCES Dept,
                      FOREIGN KEY (eid) REFERENCES Emp,
                      ON DELETE CASCADE)

CREATE TABLE Dept (   did          INTEGER,
                      buget        REAL,
                      managerid    INTEGER ,
                      PRIMARY KEY (did),
                      FOREIGN KEY (managerid) REFERENCES Emp,
                      ON DELETE SET NULL)
```

Now, we can define statements to delete employees who make more than one of their managers:

```
DELETE
FROM    Emp E
WHERE   E.eid IN (  SELECT W.eid
                    FROM   Work W, Emp E2, Dept D
                    WHERE  W.did = D.did
                    AND     D.managerid = E2.eid
                    AND     E.salary > E2.salary )
```