6.5
Integrity Contraints:

Foreign Key	Referencing Relation
PREREQUISITE.(CourseNumber)	Course.(CourseNumber)
PREREQUISITE.(PrerequisiteNumber)	Course.(CourseNumber)
SECTION.(CourseNumber)	Course.(CourseNumber)
GRADE_REPORT.(StudentNumber)	STUDENT.(StudentNumber)
GRADE_REPORT.(SectionIdentifier)	SECTION.(SectionIdentifier)

SQL Statements:

CREATE TABLE STUDENT (Name VARCHAR(30) NOT NULL, StudentNumber INTEGER NOT NULL, Class CHAR NOT NULL, Major CHAR(4), PRIMARY KEY (StudentNumber));

CREATE TABLE COURSE (CourseName VARCHAR(30) NOT NULL, CourseNumber CHAR(8) NOT NULL, CreditHours INTEGER, Department CHAR(4), PRIMARY KEY (CourseNumber), UNIQUE (CourseName));

CREATE TABLE PREREQUISITE (CourseNumber CHAR(8) NOT NULL, PrerequisiteNumber CHAR(8) NOT NULL, PRIMARY KEY (CourseNumber, PrerequisiteNumber), FOREIGN KEY (CourseNumber) REFERENCES COURSE (CourseNumber), FOREIGN KEY (PrerequisiteNumber) REFERENCES COURSE (CourseNumber));

CREATE TABLE SECTION (Sectionldentifier INTEGER NOT NULL, CourseNumber CHAR(8) NOT NULL, Semester VARCHAR(6) NOT NULL, Year CHARM NOT NULL, Instructor VARCHAR(15) PRIMARY KEY (Sectionldentifier), FOREIGN KEY (CourseNumber) REFERENCES COURSE (CourseNumber));

CREATE TABLE GRADE REPORT (StudentNumber INTEGER NOT NULL, Sectionldentifier INTEGER NOT NULL, Grade CHAR, PRIMARY KEY (StudentNumber, Sectionldentifier), FOREIGN KEY (StudentNumber) REFERENCES STUDENT (StudentNumber), FOREIGN KEY (Sectionidentifier) REFERENCES SECTION (Sectionldentifier));

6.7
REJECT should be chosen, since it will not permit automatic changes to happen that may be unintended.
BOOK AUTHORS.(Bookld) --> BOOK.(Bookld)
So now CASCADE on both DELETE or UPDATE BOOK.(PublisherName) --> PUBLISHER.(Name)
REJECT on DELETE (we should not delete a PUBLISHER tuple which has existing BOOK tuples that reference the PUBLISHER)

CASCADE on UPDATE (if a PUBLISHER's Name is updated, the change should be propagated automatically to all referencing BOOK tuples) BOOK LOANS.(Bookld) --> BOOK.(Bookld)

CASCADE on both DELETE or UPDATE (if a BOOK is deleted, or the value of its Bookld is updated (changed), the deletion or change is automatically propagated to the referencing BOOK LOANS tuples) BOOK COPIES.(Bookld) --> BOOK.(Bookld) CASCADE on both DELETE or UPDATE
if a BOOK is deleted, or the value of its Bookld is updated (changed), the deletion or change is automatically propagated to the referencing BOOK COPIES tuples.
BOOK LOANS.(CardNo) --> BORROWER.(CardNo)

CASCADE on both DELETE or UPDATE (if a BORROWER tuple is deleted, or the value of its CardNo is updated (changed), the deletion or change is automatically propagated to the referencing BOOK LOANS tuples)
Here choose REJECT on DELETE, with the idea that if a BORROWER is deleted, it is necessary first to make a printout of all BOOK_LOANS outstanding before deleting the BORROWER; in this case, the tuples in BOOK_LOANS that reference the BORROWER being deleted would first be explicitly deleted after making the printout, and before the BORROWER is deleted.
BOOK COPIES.(Branchld) --> LIBRARY BRANCH.(Branchld)
CASCADE on both DELETE or UPDATE
(if a LIBRARY BRANCH is deleted, or the value of its Branchld is updated (changed), the deletion or change is automatically propagated to the referencing BOOK COPIES tuples) BOOK LOANS.(Branchld) --> LIBRARY BRANCH.(Branchld) CASCADE on both DELETE or UPDATE (if a LIBRARY BRANCH is deleted, or the value of its Branchld is updated (changed), the deletion or change is automatically
propagated to the referencing BOOK_LOANS tuples)
here one could also choose REJECT on DELETE.

6.8
CREATE TABLE BOOK (Bookld CHAR(20) NOT NULL, Title VARCHAR(30) NOT NULL, PublisherName VARCHAR(20), PRIMARY KEY (BookId), FOREIGN KEY (PublisherName) REFERENCES PUBLISHER (Name) ON UPDATE CASCADE);

CREATE TABLE BOOK AUTHORS (Bookld CHAR(20) NOT NULL, AuthorName VARCHAR(30) NOT NULL, PRIMARY KEY (Bookld, AuthorName), FOREIGN KEY (Bookld) REFERENCES BOOK (Bookld) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE PUBLISHER (Name VARCHAR(20) NOT NULL, Address VARCHAR(40) NOT NULL, Phone CHAR(12), PRIMARY KEY (Name));

CREATE TABLE BOOK COPIES (Bookld CHAR(20) NOT NULL, Branchld INTEGER NOT NULL, No Of Copies INTEGER NOT NULL, PRIMARY KEY (Bookld, BranchId), FOREIGN KEY (Bookld) REFERENCES BOOK (Bookld)
ON DELETE CASCADE ON UPDATE CASCADE,FOREIGN KEY (Branch Id) REFERENCES BRANCH (Branch Id ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE BORROWER (CardNo INTEGER NOT NULL, Name VARCHAR(30) NOT NULL, Address VARCHAR(40) NOT NULL, Phone CHAR(12), PRIMARY KEY (CardNo));

CREATE TABLE BOOK LOANS (CardNo INTEGER NOT NULL, Bookld CHAR(20) NOT NULL,
Branchld INTEGER NOT NULL, DateOut DATE NOT NULL,
DueDate DATE NOT NULL, PRIMARY KEY (CardNo, Bookld, Branchld), FOREIGN KEY (CardNo)
REFERENCES BORROWER (CardNo) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY
(Branchld) REFERENCES LIBRARY BRANCH (Branch Id) ON DELETE CASCADE ON UPDATE
CASCADE,
FOREIGN KEY (Bookld) REFERENCES BOOK (Bookld) ON DELETE CASCADE ON UPDATE
CASCADE);

CREATE TABLE LIBRARY BRANCH (Branchld INTEGER NOT NULL, BranchName VARCHAR(20)
NOT NULL, Address VARCHAR(L1-0) NOT NULL,
PRIMARY KEY (Branchld));

6.12
a)
Select Name from Student where Major = 'CS'

b)
Select Course name
from Course, Section
where Course.Course number = Section.Course number and instructor = "king" AND (Year=07'
or Year=08)

c)
Select Course number, Semester, Year, Count(g.Student number) As "# Students" from Section
s, Grade Report g where &instructor= "king" and s.Section identifier = g.Section identifier Group
by Course_number, Semester, Year

d)
Select St.Name, C.Course name, C.Course number, C.Credit hours, S.Semester, S.Year, G.Grade
from Student St, Course C, Section 5, Grade Report G

7.5
a)
SELECT DNAME, COUNT (*) FROM DEPARTMENT, EMPLOYEE WHERE DNUMBER=DNO GROUP
BY DNAME HAVING AVG (SALARY) > 30000

DNAME	DNUMBER	COUNT(*)
Research	5	4
Administer	4	3
Headquarters	1	1

b)
SELECT DNAME, COUNT (*) FROM DEPARTMENT, EMPLOYEE WHERE DNUMBER=DNO AND SEX=M AND DNO IN (SELECT DNO FROM EMPLOYEE GROUP BY DNO HAVING AVG (SALARY) > 30000) GROUP BY DNAME

DNAME	DNUMBER	COUNT(*)
Research	5	3
Administer	4	1
Headquarters	1	1

7.7
a)
SELECT LNAME FROM EMPLOYEE WHERE DNO = (SELECT DNO FROM EMPLOYEE WHERE SALARY = (SELECT MAX(SALARY) FROM EMPLOYEE))

b)
SELECT LNAME FROM EMPLOYEE WHERE SUPERSSN IN (SELECT SSN FROM EMPLOYEE WHERE SUPERSSN = '888665555')

c)
SELECT LNAME FROM EMPLOYEE WHERE SALARY > 10000 + (SELECT MIN(SALARY) FROM EMPLOYEE)

7.8
a)
CREATE VIEW MANAGER INFO AS SELECT Dname, Fname AS Manager First name, Salary FROM DEPARTMENT, EMPLOYEE WHERE DEPARTMENT.Mgr ssn = EMPLOYEE.Ssn
b)
CREATE VIEW EMPLOYEE INFO AS SELECT e.Fname AS Employee first name, e.Minit AS Employee middle init, e.Lname AS Employee last name, s.Fname AS Manager fname, s.Minit AS Manager minit, s.Laname AS Manager Lname, Salary
FROM EMPLOYEE e, EMPLOYEE s, DEPARTMENT d WHERE e.Super ssn = s.Ssn AND e.Dno = d.Dnumber

c)
CREATE VIEW PROJECT INFO AS SELECT Pname, Dname, COUNT(WO.Essn), SUM(WO.Hours) FROM PROJECT P, WORKS ON WO,DEPARTMENT D WHERE P.Dnum = D.Dnumber AND P.Pnumber = WO.Pno GROUP BY Pno

d)
CREATE VIEW PROJECT INFO AS SELECT Pname, Dname, COUNT(WO.Essn), SUM(WO.Hours) FROM PROJECT ID, WORKS ON 1I ,DEPART D WHERE P.Dnum = D.Dnumber AND P.Pnumber = WO.Pno GROUP BY Pno HAVING COUNT(WO.Essn) > 1