Homework #4

1. Suppose that each of the following Update operations is applied directly to the database state shown in Figure 5.6. Discuss all integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints.
   a. Insert <'Robert', 'F', 'Scott', '943775543', '1972-06-21', '2365 Newcastle Rd, Bellaire, TX', M, 58000, '888665555', 1> into EMPLOYEE.
      This insertion satisfies all constraints, so it is acceptable
   b. Insert <'ProductA', 4, 'Bellaire', 2> into PROJECT
      This insertion violates the referential integrity constraints specified on Dnum because no DEPARTMENT tuple exists with DNUMBER = 2
      A way to enforce this constraint is to insert a DEPARTMENT tuple with Dnumber=2 and then perform the above insertion operation. Else, change the DNUMBER of the above tuple to valid Dnumber such as 5, 4 or 1
   c. Insert <'Production', 4, '943775543', '2007-10-01'> into DEPARTMENT
      This insertion violates the key constraint because another tuple (i.e Administration) with the same Dnumber already exists
      A way to enforce this constraint is to change the Dnumber for Production to one that is not being used in the Department relation
   d. Insert <'677678989', NULL, '40.0'> into WORKS_ON
      This insertion violates the entity integrity constraint (null for the primary key Pno)
      It also violates referential integrity specified on SSN because no EMPLOYEE tuple exists where SSN = '677678989'
      A way to enforce this constraint is to change the null value to valid Pno, also insert an employee tuple with SSN = '677678989' and then perform the above insert operation.
      Another option to enforce this constraint is to change the ESSN to a valid SSN from the EMPLOYEE table such as 666884444
   e. Insert <'453453453', 'John', 'M', '1990-12-12', 'spouse'> into DEPENDENT.
      This insertion satisfies all constraints, so it is acceptable
   f. Delete the WORKS_ON tuples with Essn = '333445555'
      This deletion is acceptable
   g. Delete the EMPLOYEE tuple with Ssn = '987654321'
      This deletion is not acceptable because tuples in EMPLOYEE, DEPARTMENT, WORKS_ON and DEPENDENT refer this tuple. Hence, if the tuple is deleted, referential integrity violations will result
      A way to enforce this constraint is to delete tuples that reference the tuple that is being deleted. Another option is to modify the referencing attribute to another value such as null or valid tuple value
   h. Delete the PROJECT tuple with Pname = 'ProductX'
      This deletion is not acceptable because tuples in WORKS_ON refer to this tuple. Hence referential integrity violations will result.
      A way to enforce this constraint is to delete tuples that reference the tuple that is being deleted. Another option is to modify the referencing attribute to another value such as null or valid tuple value

i. Modify the Mgr_ssn and Mgr_start_date of the DEPARTMENT tuple with Dnumber = 5 to '123456789' and '2007-10-01', respectively.
   This modification satisfies all constraints, so it is acceptable
j. Modify the Super_ssn attribute of the EMPLOYEE tuple with Ssn = '999887777' to '943775543'.
   This modification satisfies all constraints, so it is acceptable
k. Modify the Hours attribute of the WORKS_ON tuple with Essn = '999887777' and Pno = 10 to '5.0'.
   This modification satisfies all constraints, so it is acceptable

2. Airline reservation system:
   - Give the operations for this update.
     Assuming that vacant seats are available on Flight or flight leg asked by customer on the specified date, update operation can be: (let flight number – FL1, leg number – 1, date – '2018-04-24', Seat number allocated – 2, Customer name – John and Phone number – 555555555)

     Insert<'FL1', '1', '2018-04-24', '2', 'John', '555555555'> into SEAT_RESERVATION

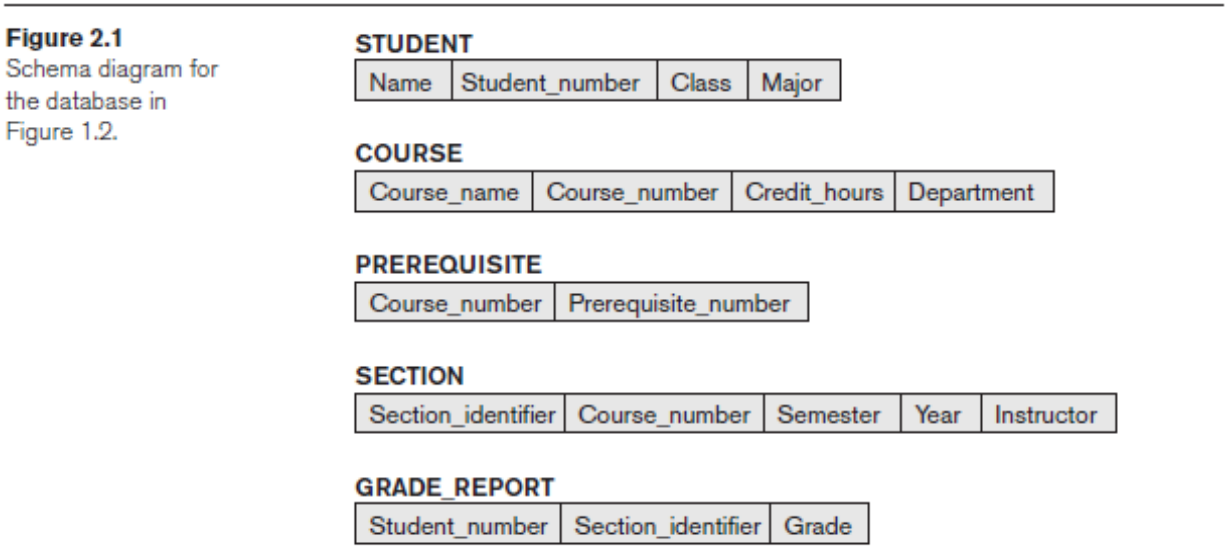
   - What types of constraints would you expect to check?
     The following constraints need to be met
     a. Asked flight number or flight leg is available on a given date. Date can be checked from LEG_INSTANCE table
     b. A non-reserved seat must exist for specific date and flight. We can get total number of seats available from AIRPLANE

   - Which of these constraints are key, entity integrity, and referential integrity constraints, and which are not?
     Referential Integrity constraint – Asked flight number or flight leg is available on a given date. Data can be checked from LEG_INSTANCE table
     Entity integrity constraint – A non-reserved seat must exist for specified date and flight and leg number so that there is unique SEAT_RESERVATION schema

   - Specify all the referential integrity constraints that hold on the schema shown in Figure 5.8.
     FK – Foreign key
     Foreign keys in given schema are:
     a. Flight number of FLIGHT_LEG is FK of FLIGHT
     b. Departure_airport_code and Arrival_airport_code of LEG_INSTANCE are FK for AIRPORT
     c. Departure_airport_code and Arrival_airport_code of FLIGHT are FK for AIRPORT
     d. Airport_code of CAN_LAND are FK for AIRPORT
     e. Flight_number and Leg_number of LEG_INSTANCE are FK for FLIGHT_LEG
     f. Airplane_id of LEG_INSTANCE is FK for FLIGHT

g. Flight_number of FARE is FK for FLIGHT
h. Flight_number of SEAT_RESERVATION is FK for FLIGHT
i. Flight_number of LEG_INSTANCE is FK for FLIGHT
j. Flight_number, Leg_number and Date of SEAT_RESERVATION are FK for FLIGHT
k. Airplane_type_name of CAN_LAND is FK of AIRPLANE_TYPE

3. Figure:



DDL statements for above schema

- CREATE TABLE STUDENT(Name VARCHAR(30) NOT NULL, Student_number INTEGER NOT NULL, Class CHAR NOT NULL, Major CHAR(4), PRIMARY_KEY(Student_number));
- CREATE TABLE COURSE(Course_name VARCHAR(30) NOT NULL, Course_number CHAR(8) NOT NULL, CreditHours INTEGER, Department CHAR(4), PRIMARY_KEY(Course_number), UNIQUE(Course_number));
- CREATE TABLE PREREQUISITE(Course_number CHAR(8) NOT NULL, PrequisiteNumber CHAR(8) NOT NULL, PRIMARY_KEY(Course_number, PrequisiteNumber), FOREIGN_KEY (Course_number) REFERENCES COURSE(CourseNumber), FOREIGN_KEY (PrequisiteNumber) REFERENCES COURSE(CourseNumber));
- CREATE TABLE SECTION(Section_identifier INTEGER NOT NULL, Course_number CHAR(8) NOT NULL, Semester VARCHAR(6) NOT NULL, Year CHAR(4) NOT NULL, Instructor VARCHAR(15), PRIMARY_KEY(Section_identifier), FOREIGN_KEY (Course_number) REFERENCES COURSE(CourseNumber));
- CREATE TABLE GRADE_REPORT(Student_number INTEGER NOT NULL, Section_identifier INTEGER NOT NULL, Grade CHAR, PRIMARY_KEY(Student_number, Section_identifier), FOREIGN_KEY (Student_number) REFERENCES STUDENT (Student_number), FOREIGN_KEY (Section_identifier) REFERENCES SECTION (Section_identifier));

4. SQL queries:
   - Retrieve the names of all employees in department 5 who work more than 10 hours per week on the ProductX project.

Select emp.Fname, emp.Lname from employee emp, works_on w, project p where emp.Dno = 5 and emp.ssn = w.Essn and w.Pno = p.pnumber and p.pname = 'ProductX' and w.hours>10

Result:
John Smith
Joyce English

- List the names of all employees who have a dependent with the same first name as themselves
  select emp.Fname, emp.Lname from employee emp, dependent d where emp.ssn = d.essn and emp.Fname = d.Depandent_name

  Result: (Empty)

- Find the names of all employees who are directly supervised by 'Franklin Wong'.
  Select emp.Fname, emp.Lname from employee emp, employee emp1 where emp1.Fname = 'Franklin' and emp1.Lname = 'Wong' and emp.superssn = emp1.ssn

  Result:
  John Smith
  Ramesh Narayan
  Joyce English

5. Specify the following queries in SQL on the database schema of Figure 1.2.
   a. Retrieve the names of all senior students majoring in 'cs' (computer science).
      Select name from student where Major = 'CS'

   b. Retrieve the names of all courses taught by Professor King in 2007 and 2008.
      Select course_name from course, section where course.course_number = section.course_number and instructor = 'King' and (Year='07' or Year ='08')

   c. For each section taught by Professor King, retrieve the course number, semester, year, and number of students who took the section.
      Select course_number, semester, year, count(g.student_number) as student_count from section s, grade_report g where s.instructor='King' and s.section_identifier=g.section_identifier group by course_number, semester, year

   d. Retrieve the name and transcript of each senior student (Class = 4) majoring in CS. A transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.
      Select st.name, c.course_name, c.course_number, c.credit_hours, s.semester, s.year, g.grade from student st, course c, section s, grade_report g where class = 4 and Major = 'CS'

```
and st.student_number = g.student_number and g.section_identifier = s.section_identifier
and s.course_number = c.course_number;
```