DBMS Theory Assignment#1

Submission Date: 11.02.2019

1. Consider the following relations in a university database:

Student (roll, name, dept, batch, email)

Course (courseno, cname, offering dept, semester no, course type, credit)

Prerequisite(courseno, prcourseno)

Registration (roll, courseno, semester no)

Grades (roll, courseno, grade)

Attendance (courseno, roll, date, status);

The course type can be core-theory, core-lab, elective. Semester no can be 1 to 8. Grade for any course can be 5 to 10. The status can be P for present and A for absent. Roll and courseno should be assigned through sequence.

Write efficient implementation for the following tasks:

(i) Students register for different courses. A student is not allowed to register for more than 6 courses in a semester and/or with credit more than 24. A student is not allowed to register for a core course unless he/she scores at least 6 in all the pre-requisite courses for that course. For registering in elective courses, the student must have register in the pre-requisite courses.

(ii) Raise automatic warning for the students whose attendance falls below 80% at every five class interval against a course and store this data in a new table called attendance_warning for each such student. In case, there is 2 such warning against a student, the student is de-registered from the course. The de-registration data with date to be automatically stored in a separate table and the related entry to be deleted from the Registration Table.

(iii) Display the student-wise total grade for each semester considering all the subjects. The total grade will be computed as the credit-grade weighted average of the courses in a semester.

(iv) Display the batch-wise toppers for each semester

(v) Display the student-wise degree grade considering grades of all 8 semesters. The total grade will be computed as the average of the grades in all semester.

(vi) Display batch wise degree-toppers considering all 8 semester

(vii) Display the dept-wise student count with degree grade more than 9 for a batch.

(viii) Change the name of any course through views.

2. Consider the following relations in an organization database:

Emp(eno, ename, sal, comm, dept-no, job type, joining_date)

Dept(dno, dname, location, dmgrno)

Manpower_Budget(dno, year, cost)

Job type can be SOFTWARE ENGINEER, HR, PRODUCION ENGINEER, RESEARCH SCIENTIST, DELIVERY ENGINEER.

Write efficient implementation for the following tasks:

(i) After every year of job completion salary of the employees will be incremented as follows:
SE – 10% (If manager 15%)
HR – 7% (If manager 10%)
PE – 15% (If manager 18%)
DE – 12% (if manager 15%)
RS – 15% (if manager 18%)

However, the total increment for every year should be less than the cost assigned for manpower for that year. In violation of this warning should be raised. For each successful salary increase, an entry is to be added in a new table called salary_hike with eno, ename, date of increment, increment amount.

(ii) Display the name of top 3 salary earners in each dept-no

(iii) Change the dept-location through views

(iv) Whenever a new employee joins/leaves the organization, automatically insert his details with date of joining, last-salary drawn, date of leaving in a separate table called employee_history.

3. Consider the following relations for an OPD in an hospital:
Patient(patient-id, patient-name, DOB, Sex)
Doctor(Doctor-id, Name, specialization, Unit)
OPD_Schedule(Doctor-id, date, time, fees)
Appointment (appointment-no, patient-id, doctor-id, date)
OPD_payments(appointment-no, patient-id, amt, date_payment)

Write efficient implementation for the following tasks:

(i) Display the list of patients who has visited the same doctor in the hospital more than 2 times during a given time-period

(ii) The records in OPD_payments to be automatically updated based on Schedule and Appointment table. If a patient visits twice the same doctor within a week, amt in OPD_payments will be automatically set to 0. For senior citizen female patients, the fees to be reduced by 50% and entry should be automatically inserted in OPD_payments table.

(iii) Display the number of patients visited to the doctors unit-wise and date-wise.

(iv) Change the date and time (not other field) in OPD_Schedule of any given doctor through views

(v) Create a OPD log including patient name, age, date_of_visits and doctor names for a given period.

(vi) Increase the fees of a given doctor through views.

4. Consider the following database:

Student(Sroll, Name, Branch, Batch, Programme)
Course(CID, Cname, Instructor Name)
Attendance(Sroll, Course ID, Period, Total#class, Attendance)

Write efficient implementation for the following tasks:

(i) Display the name of students whose attendance is below 80% in a given course.

(ii) Create a view to list attendance records of the students in all subjects for a given batch.

(iii) Use the view in (ii) to list the student names whose attendance is below 70% in all the subjects.

(iv) Create a view to display attendance of students for a given course ID.

(v) Use this view in (iv) to update the attendance of a student (assume the update is done by a authorized user)

(vi) Assume a student is deregistered from a given course: Delete the record of the student from Student table - his/her data to be deleted from the attendance table.

(vii) Internal marks for attendance for a student is to be calculated against a course as per the following rules:

Attendance %	Marks
>=95	5
85-95	4
75-85	3
60-75	2
<60	0

Write a PL/SQL function which takes sroll and course id as input and computes the attendance percentage for that course and returns the marks for that course. Then, Insert the Sroll and Marks in a newly created table called ATT_Marks(Sroll, Marks). This complete task is to be done in a single PL/SQL program.

Submission Guidelines:

Submit SQL and PL-SQL codes in a file. Show the execution in person to the assigned TA (Bata Krishna Tripathy).