**DBMS Lab 2017-18 Spring Semester**

**Lab Test 1 (February 6, 2018) – 100 Marks (ODD PC NOS)**

1. Consider that we want to design a database that will store details of books in a library. The following information is given.

Each book is identified by a book\_id. We need to maintain the book\_name and topic for each book. We also need to maintain data about all the authors who have written a book. Each author is identified by an author\_id. Other attributes for author are author\_name and date\_of\_birth. The library is only concerned with authors who have written at least one book present in the library and similarly, it is concerned about only those books that it has. There might be multiple editions for a book like 1, 2, 3, etc. We need to maintain the year of publication for each book edition and the corresponding price (Assume that the first edition has edition number 1). Library members are identified by the attribute member\_id. Other attributes of member are member\_name, one or more phone numbers and one or more degrees (B.Tech., M.Tech, Ph.D., etc.). Members can borrow any edition of any book. For each such borrowing, we need to maintain date\_of\_issue, due\_date\_of\_return and fine paid.

Draw an Entity-relationship (E-R) diagram on the piece of paper provided using Chen’s notation (NOT Crow’s feet notation) for capturing the above-mentioned information, clearly identifying all the entity sets, relationship sets, cardinalities, participation and attributes (including the type of attribute like key attribute, multi-valued attribute, derived attribute, etc.). A simplifying assumption is that the library has only one copy of each edition of any given book. You need to take care of the following conditions: There is at least one author for each book. The same set of authors would write various editions of the same book, each member can borrow a particular edition of a book only once during the entire course of time, each member can borrow upto 4 books at any given point of time. Members may want to query about books written by various authors. State all other assumptions that you make. They should, however, be reasonable.

From the ER-Diagram, derive the tables using correct rules for relational model generation from ER model. There should not be any redundant table and also there should not be redundant information in each table. Clearly mark the columns, primary keys and foreign keys for each table. This forms your database schema. Draw the database schema on the piece of paper provided. Write appropriate CREATE table SQLs to create these tables in the database. You must specify appropriate data types, keys and constraints.

Submit hardcopies of the (a) ER-Diagram and (b) Database Schema. Upload a text file containing your CREATE table SQLs against **Link\_1 for odd question**. Name it as: CREATE\_<Roll\_no>.sql. The text file should contain a header having your roll\_no, name and PC No. **[15+15+10=40]**

2. Consider that we have the following tables in a database (primary keys are underlined):

**Student** (roll\_no int, student\_name varchar(30) not null, year\_of\_admission int, dept\_cd char(2) not null, cgpa decimal(5,2), percentage\_marks decimal(5,2), hall\_cd char(2))

**Course** (course\_cd char(5), course\_name varchar(30) not null, credits int not null, max\_marks decimal(5,2) not null, dept\_cd char(2) not null)

**Department** (dept\_cd char(2), dept\_name varchar(30) not null, hod\_name varchar(30) not null, year\_of\_establishment int)

**Hall** (hall\_cd char(2), hall\_name varchar(30) not null, no\_of\_rooms int not null)

**Registration** (roll\_no int, course\_cd char(5), grade\_point int, marks\_obtained decimal(5,2))

[Here valid values of grade point are numbers like 10, 9, 8, etc. (they correspond to grades EX, A, B, etc., but we are not maintaining the grades) and valid value of marks is the absolute marks obtained.]

Write SQLs to satisfy the following database operations (**Each should be a single SQL.** **At most for one SQL, you are allowed to use a view definition**). **[6×10=60]**

1. Update percentage\_marks in Student table based on marks\_obtained and max\_marks

update student s set s.percentage\_marks=(select (sum(r.marks\_obtained)/sum(c.max\_marks))\*100 from Registration r, Course c where c.course\_cd=r.course\_cd and r.roll\_no=s.roll\_no);

|  |  |
| --- | --- |
| Before Update | After Update |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Aditya | 2013 | CS | 9.36 | 89 | AB | | Adam | 2012 | CS | 8.83 | 87 | AB | | Anil | 2012 | CS | 8.4 | 81.67 | AB | | Anoop | 2014 | CS | 7.38 | 69.5 | AB | | Aravind | 2016 | EC | 8.57 | 81 | AB | | Bunny | 2014 | CS | 9.14 | 87.67 | AB | | Bhanu | 2014 | CS | 7.29 | 65 | AB | | Cummins | 2014 | MA | 7.21 | 65 | AB | | Saptarshi | 2014 | EC | 7.29 | 67.84 | EF | | Sabarish | 2014 | MA | 8.17 | 76.88 | AB | | Jaishree | 2014 | MA | 8.5 | 82.73 | AB | | Jai | 2013 | CS | 8 | 74.4 | AB | | Debopriyo | 2014 | EC | NULL | NULL | EF | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Aditya | 2013 | CS | 9.36 | 88.33 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Adam | 2012 | CS | 8.83 | 87 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Anil | 2012 | CS | 8.4 | 81.67 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Anoop | 2014 | CS | 7.38 | 69.5 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Aravind | 2016 | EC | 8.57 | 81 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Bunny | 2014 | CS | 9.14 | 87.67 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Bhanu | 2014 | CS | 7.29 | 65 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Cummins | 2014 | MA | 7.21 | 65 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Saptarshi | 2014 | EC | 7.29 | 67.84 | EF |  |  |  |  |  |  |  |  |  |  |  |  | | Sabarish | 2014 | MA | 8.17 | 76.88 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Jaishree | 2014 | MA | 8.5 | 82.73 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Jai | 2013 | CS | 8 | 74.4 | AB |  |  |  |  |  |  |  |  |  |  |  |  | | Debopriyo | 2014 | EC | NULL | NULL | EF |  |  |  |  |  |  |  |  |  |  |  |  | |

1. Insert a row in Hall table where hall\_cd = AB, hall\_name = ABC and no\_of\_rooms is equal to the total number of students belonging to the departments whose names are five or more characters long and have ‘A’ as the second character.

insert into Hall values('AB','ABC',(select count(\*) from student s where s.dept\_cd in (select dept\_cd from Department where dept\_name like '\_A\_\_\_%')));

|  |  |
| --- | --- |
| Before Insert | After Insert |
| |  |  |  | | --- | --- | --- | | EF | EFG | 12 | | PR | PRS | 15 | | UV | UVW | 0 | | XY | XYZ | 12 | | |  |  |  | | --- | --- | --- | | EF | EFG | 12 | | PR | PRS | 15 | | UV | UVW | 0 | | XY | XYZ | 12 | | AB | ABC | 3 | |

1. Delete all rows from the Hall table for which the no\_of\_rooms is less than the number of students in the Student table for whom there is NULL value in the hall\_cd column

delete from Hall where no\_of\_rooms < (select count(\*) from student where hall\_cd is NULL);

Query OK, 1 row affected (0.02 sec)

Hall with hall\_cd ‘UV’ should be deleted.

|  |  |
| --- | --- |
| Before Delete | After Delete |
| |  |  |  | | --- | --- | --- | | EF | EFG | 12 | | PR | PRS | 15 | | UV | UVW | 0 | | XY | XYZ | 12 | | AB | ABC | 3 | | |  |  |  | | --- | --- | --- | | EF | EFG | 12 | | PR | PRS | 15 | | XY | XYZ | 12 | | AB | ABC | 3 | |

1. List roll\_no, student\_name for all students with the highest cgpa

select roll\_no,student\_name from student where cgpa=(select max(cgpa) from student);

+---------+--------------+

| roll\_no | student\_name |

+---------+--------------+

| 10000 | Aditya |

+---------+--------------+

1. List roll\_no, student\_name for all CS students (i.e., students with dept\_cd = CS) who have not taken any subject offered by the CS department (i.e., subjects with dept\_cd = CS)

select roll\_no,student\_name from student s where s.dept\_cd='CS' and 0=(select count(\*) from Registration r where r.roll\_no=s.roll\_no and (select c.dept\_cd from Course c where c.course\_cd=r.course\_cd)='CS');

+---------+--------------+

| roll\_no | student\_name |

+---------+--------------+

| 10002 | Anil |

| 10010 | Shabharesh |

+---------+--------------+

select roll\_no,student\_name from student s where s.dept\_cd='CS' and not exists (select \* from Registration r, Course c where r.roll\_no=s.roll\_no and r.course\_cd = c.course\_cd and c.dept\_cd=’CS’);

1. List dept\_cd, dept\_name for all departments established before 1990 that offer more than 5 courses.

select d.dept\_cd,d.dept\_name from Department d where d.year\_of\_establishment<1990 and (select count(\*) from Course c where c.dept\_cd=d.dept\_cd)>5;

+---------+------------------+

| dept\_cd | dept\_name |

+---------+------------------+

| CS | Computer Science |

+---------+------------------+

1. List course\_cd, course\_name for all courses which have more number of students than the average number of students in all other courses

select c.course\_cd,c.course\_name from Course c where (select count(\*) from Registration r1 where r1.course\_cd=c.course\_cd) > (select count(\*)/(select count(\*)-1 from Course) from Registration r2 where r2.course\_cd<>c.course\_cd);

+-----------+-----------------------+

| course\_cd | course\_name |

+-----------+-----------------------+

| CS105 | CN |

| CS106 | CNLab |

| CS107 | Information Retrieval |

| CS108 | Deep Learning |

| CS109 | Seminar |

| EC000 | MicroElectronics |

| EC001 | DLD |

| MA000 | AdvancedMaths |

+-----------+-----------------------+

1. List roll\_no, student\_name for all students who have the 6th highest percentage\_marks

select roll\_no,student\_name from student s1 where 5 = (select count(distinct(percentage\_marks)) from student s2 where s1.percentage\_marks < s2.percentage\_marks);

+---------+--------------+

| roll\_no | student\_name |

+---------+--------------+

| 10004 | Aravind |

+---------+--------------+

1. List roll\_no, student\_name and hall\_name for all students who may or may not live in halls

select roll\_no,student\_name,hall\_name from student left join Hall on student.hall\_cd=Hall.hall\_cd;

+---------+--------------+-----------+

| roll\_no | student\_name | hall\_name |

+---------+--------------+-----------+

| 10000 | Aditya | ABC |

| 10001 | Adam | ABC |

| 10002 | Anil | ABC |

| 10003 | Anoop | ABC |

| 10004 | Aravind | ABC |

| 10005 | Bunny | ABC |

| 10006 | Bhanu | ABC |

| 10007 | Cummins | ABC |

| 10009 | Sabarish | ABC |

| 10011 | Jaishree | ABC |

| 10012 | Jai | ABC |

| 10008 | Saptarshi | EFG |

| 10013 | Debopriyo | EFG |

| 10010 | Shabharesh | NULL |

+---------+--------------+-----------+

14 rows in set (0.00 sec)

1. List roll\_no, student\_name for all students whose year\_of\_admission is before 2013 and who have not registered for any course

select s.roll\_no,s.student\_name from student s where s.year\_of\_admission<2013 and not exists(select \* from Registration r where r.roll\_no=s.roll\_no);

Empty set (0.00 sec)

Upload a text file containing your SQLs against **Link\_2 for odd question.** Name it as: **UPD\_SEL\_<Roll\_no>.sql.** The text file should contain a header having your roll\_no, name and PC No.

\*For example, let the percentage\_marks for roll numbers 1, 3, 4, 6, 8, 7, 2, 9, 11, 12 are 60.5, 70.7, 50.2, 90.1, 87.7, 66.7, 56.8, 34.9, 82.8, 60.5, respectively. Then, roll numbers 1 and 12 have the 6th highest percentage\_marks. Roll number 2 has the 7th highest percentage\_marks.

[Penalty for plagiarism/copying: You will be awarded 0 for all the problems for the lab day and an additional 5 marks will be deducted out of the total of 40 in Lab. All persons involved will be awarded the same penalty irrespective of who has copied from whom]