Sai University

Data Base Management Systems

Mid-Semester Examination (Make-Up), Semester A, 2023–2024

Date	: 25 October, 2023	Marks	: 20
Duration	: 90 minutes	Weightage	: 20%

1. Consider a university database with relations

   - $student(sid, name)$ — student ID and name for each student
   - $course(cid, title)$ — course ID and title for each course
   - $passed(sid, cid)$ — which courses each student has cleared already, in terms of student and course IDs
   - $core(cid)$ — IDs of core courses

   Write relational algebra expressions to compute each of the following relations:

   (a) IDs of students who have not yet cleared *any* course *(2 marks)*

   (b) IDs of students who have not yet cleared all core courses *(2 marks)*

   (c) IDs of core courses that are still pending for at least one student *(2 marks)*

2. For the ongoing cricket World Cup, a database is being maintained that includes the following two tables:

   Players(PlayerID, Name, Country)
   Runs(MatchNumber, PlayerID, Runs)

   Each country has a list of players registered for the World Cup and *PlayerID* is a unique ID assigned to each registered player. The competition consists of 48 matches and each match is identified by its *MatchNumber*, from 1 to 48.

   The *Players* table lists all the registered players. The *Runs* table records the runs scored in all the matches played so far, and is updated after every match. For each match, the table records runs for those players who actually batted. There is no entry for a player who did not play in the match or who played but did not get a chance to bat.

   Whenever a player comes to bat, the screens at the stadium show the runs that the player has scored so far in this World Cup. To compute this quantity, we need "join" the two tables above to create a table of the form

   Aggregate(PlayerID, Name, TotalRunsTillNow)

   that records the runs scored by *all* registered players, including those who have yet to bat in any match.

   (a) Explain why a natural join is not adequate for this purpose. *(2 marks)*

   (b) What kind of join should we use to ensure that *every* player is included in the table *Aggregate*? *(2 marks)*

   (c) Write an SQL query to compute the table *Aggregate*. *(4 marks)*

3. A bank's database contains a table *Accounts(CustomerID, AccountNo, BranchID)*, where *CustomerID* is a unique ID for each customer, *AccountNo* is the bank account number and *BranchID* is a unique ID for each branch of the bank.

   Account numbers are unique across the bank and each account number is attached to one branch of the bank. The bank permits joint accounts, with more than one customer associated with an account. A customer may have many accounts in the bank, but is restricted to at most one account in each branch, whether it is single or joint.

   (a) What functional dependencies can you infer from these constraints? *(2 marks)*

   (b) Compute a BCNF decomposition of the table *Accounts*. *(2 marks)*

   (c) Explain whether your BCNF decomposition is dependency preserving. *(2 marks)*