DATABASE TECHNOLOGY
WEEK 11 ASSIGNMENT

1.Explain each type of anomaly with its examples.

Anomalies are problems that can occur in poorly planned, unnormalized databases where all the data is stored in one table (a flatfile database).

Types Of Anomalies:
The three type of anomalies that can arise in the database because of redundancy are
1. Insertion
2. Deletion
3. Modification/ Updation anomalies.

EXAMPLE:
Consider a relation emp_dept with attributes:
1. E# {with the primary key as E#.}
2. Ename
3. Address
4. D#
5. Dname
6. Dmgr#

Insertion anomaly: Let us assume that a new department has been started by the organization but initially there is no employee appointed for that department, then the tuple for this department cannot be inserted into this table as the E# will have NULL, which is not allowed as E# is primary key.
This kind of a problem in the relation where some tuple cannot be inserted is known as insertion anomaly.

Deletion anomaly: Now consider there is only one employee in some department and that employee leaves the organization, then the tuple of that employee has to be deleted from the table, but in addition
to that the information about the department also will get deleted.
This kind of a problem in the relation where deletion of some tuples can lead to loss of some other data
not intended to be removed is known as deletion anomaly.

Modification /update anomaly: Suppose the manager of a department has changed, this requires thatthe Dmgr# in all the tuples corresponding to that department must be changed to reflect the new status. If we fail to update all the tuples of the given department, then two different records of employee working inthe same department might show different Dmgr# leading to inconsistency in the database.This is known as modification/update anomaly. The data redundancy can not be totally removed from the database, but there should be controlled redundancy,

For example:
Consider a relation Student_report(S#, Sname, Course#, SubjectName, Marks) to store the marks of a student for a course having some optional subjects, but all the students might not select the same optional-papers. Now the student name appears in every tuple, which is redundant and we can have two tables as
1. Students(S#, Sname, CourseName)
2. Report(S#, SubjectName, Marks).

However, if we want to print the marksheet for every student using these tables then a join operation, which is a costly operation, in terms of resources required to carry out, has to be performed in order to get
the name of the student. So to save on the resource utilization, we might opt to store a single relation, students_report only.