1) Student Name : Venkatesh G D

2) USN : 2GI19CS175

3) BE / MBA : BE

4) Semester : 4

5) Course Name : DBMS

6) Course Code :

7) Name of Colg : BNS GIT

8) Date & Time : 20-07-21 ; 10:30 am

9) Mob Number : 9972287030

10) Signature

7) Join:

* The most useful & most common operation
* Tables are related by having columns in common; primary key on one table appears as foreign key in another
* Join uses this relatedness to combine the two tables in one
* Join is useful because both Select & Project work on only one table at a time.

Types of Join operation

1) Inner Join
   a) Theta join
   b) Equijoin
   c) Natural join
2) Outer Join:
   a) Left outer join
   b) Right outer join
   c) Full outer join

* 1) a.) Theta join: Combines tuples from different relation provided they satisfy the condition.
  Denoted by symbol θ. Notation $R_1 \bowtie_\theta R_2$
  Example:
  STUDENT $\bowtie_{Student.std = Subject.class}$ SUBJECT.
  b) Natural join: Does not use any comparison operators. Does not concatenate the way a cartesian product does.
  Example:
  STUDENT $\bowtie_{Student.std = Subject.class}$ SUBJECT
  c) Equijoin: When Theta joins only equijoin equality comparison operator, it is said equijoin

7) 2) a. Left Outer Join (R ⟕ S): All tuples from left relation, R, are included in resulting relation.

Eg:

Courses ⟕ HOD

b) Right Outer Join (R ⟖ S): All tuples from right relation, S, are included in resulting relation.

c) Full outer Join: All the tuples from both participating relations are included in resulting relation.

---

3) Need for normalization:

→ A properly normalized database should have:

* Scalar values in each fields
* Absence of redundancy
* Minimal use of null values
* Minimal loss of information.

→ Levels of Normal form are based on amount of redundancy in database

* First Normal form (1NF): A table is considered to be in 1NF if all fields contain only scalar values

* Second Normal form (2NF): For a table to be in 2NF
  - The database is in first normal form
  - All non key attributes in table must be functionally dependant on entire primary key.

3) * Third Normal form: This form dictates all non-key attributes of a table must be functionally dependant on a candidate key i.e., there can be no interdependencies among non-key attributes.
For a table to be in 3NF:
- The table should be second normal form
- No attributes is transitively dependent on primary key

1) i) Domain constraint: Defined as definition of a valid set of values for an attribute.
Example:-

ID	NAME	SEM	AGE
1000	Tommy	1	17
1001	Jim	3	17
1002	Ivy	4	A

↑
Not allowed, because AGE is int attribute.

ii) Key constraints:
- Superkey: A set of attributes SK of R such that no 2 points tuples in valid relation instance r(R) will have same value for SK.
- Key: A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

Example: The CAR relation schema:
CAR(State, RegNo, SNO, Make, Model, Year).
has 2 keys, Key1 = {State, RegNo}, Key2 = {SNO} which are also superkeys {SNO, Make} is a superkey but not key

1.) iii.) Candidate Key: in SQL is a set of attributes that uniquely identify tuples in a table. Its a super key with no repeated attributes.

Eg:

Student ID	Roll No	First Name	Last Name	email
1	11	Jimmy	Princton	abc@hotmail.
2	12	Nickey	Bright	xyz@gmail.com
3	13	Daniel	Nixon	ijk@yahoo.co.

In the given example student ID, Roll No, and email 3 candidate keys which help us uniquely identify student record in table.

iv.) Entity Integrity: The primary key attributes PK of each relation schema R in S cannot have null values in any tuple r(R). This is because primary key values are used to identify individual tuples

Eg:

R.No	Name	Sem	City
11	Ivy	2	Pune
20	Flick	3	Delhi
23	Jimm	4	Pune
NULL	Kim	5	Delhi

* Key is RNO
* Tuple since (of PK) is Null.

v.) Referential integrity: A constraint involving 2 relations

Eg:

1) Ex:

Emp	Name	Name	Age	DNo — Foreign Key	
1		Jackie	20	11	
2		Harvey	90	24	
3		Johnny	27	18	— Not allowed as D_No
4		Jim	38	13	18 is not defined as Primary key of table 2

Primary Key → DNO	DLocation
11	Delhi
24	Pune
13	Mumbai

5) → As general rule, if you ~~want~~ run across a multivalued attribute, this a major hint that you need another entity. The only way to handle multiple values of same attribute is to create an entity of which you can store multiple instances, one for each value of attribute.
In the case of employee entity we would need a dependent entity that could be related to employee entity.

→ 1NF: A table is considered to be in 1NF if all fields contains only scalar values.

2.) i.) Select *
From DEPARTMENT
where DNAME = "Mechanical Engineering Department"

ii.) Select *
From students as S, Course C, enroll e,
department d
WHERE S.SEM = 5 AND S.USN = E.USN
AND e.course # = C.course # AND C.Type = "ELECTIVE
AND S.dno = d.dno AND d.DNAME = "CSE".

iii.) SELECT d.dName. COUNT (S.USN)
FROM DEPARTMENT d, STUDENTS
GROUP BY d.DNAME;

iv.) SELECT d.DNAME
FROM DEPARTMENT d
WHERE ~~COUNT~~ ~~MARKS~~ COUNT (d.USN)

v.) SELECT COUNT (S.USN)
FROM DEPARTMENT d, STUDENT S, ENROLL E
WHERE S.SEM = 2 AND E.USN = S.USN