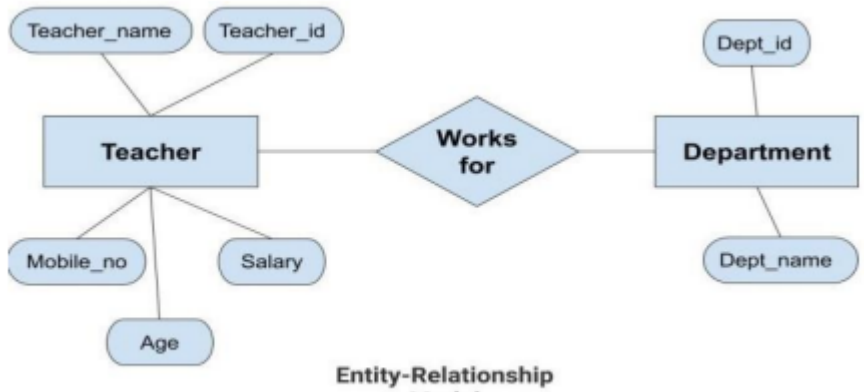
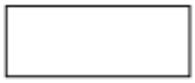
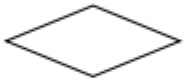
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Question Bank(R20) - III SEM- B20
(20PC0502) DATABASE MANAGEMENT SYSTEM

UNIT-1

TEN MARKS QUESTIONS:

QNo	Question	BTL
1.	**Discuss in detail about the elements of E-R model.** ● The Entity Relationship (ER) data model allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design. ● Entity-Relationship Model or simply ER Model is a high-level data model diagram. ● In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand. **Example:**  Entity-Relationship Model **Elements of E-R model:** ● There are three basic elements in an ER Diagram: entity, attribute, relationship. ● There are more elements which are based on the main elements. They are weak entity, multi valued attribute, derived attribute, weak relationship, and recursive relationship.	1

Element	Representation	Symbol
Entities	Rectangles	
Attributes	Ellipses	
Links	Lines	
Relationships	Diamonds	
Weak Entity	Double Lined Rectangle	
Multi-valued Attribute	Doubled Lined Ellipse	
Weak Relationship	Double Lined Diamond	

Entities:

- Entity is a real-world thing. It can be a person, place, or even a concept. An entity is described using a set of attributes.
- Example: Teachers, Students, Course, Building, Department, etc are some of the entities of a School Management System.

Attributes:

- An attribute is a property or characteristic of an entity. An entity may contain any number of attributes.
- One of the attributes is considered as the primary key. In an Entity-Relation model, attributes are represented in an elliptical shape.
- Example: The entity teacher has the property like teacher id, salary, age, etc.

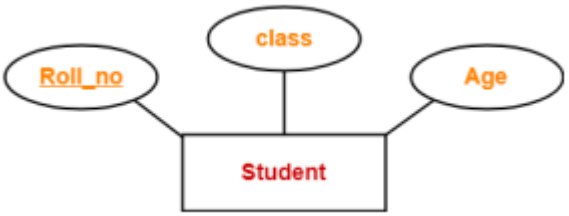
Types of Attributes:

There are six types of attributes:

- Simple attributes
- Composite attributes
- Single-valued attributes
- Multi-valued attributes
- Derived attribute
- Key attributes

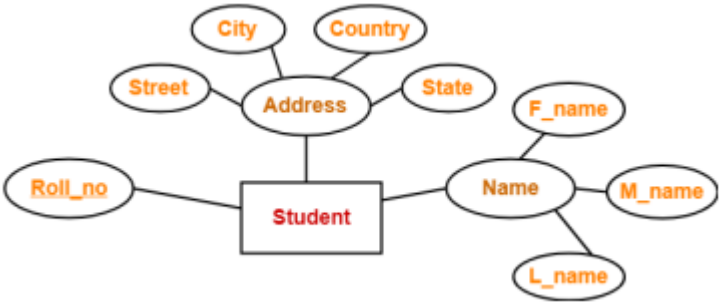
Simple attribute:

- An attribute which cannot be further subdivided into components is a simple attribute.
- Example: The roll number of a student, the id number of an employee.



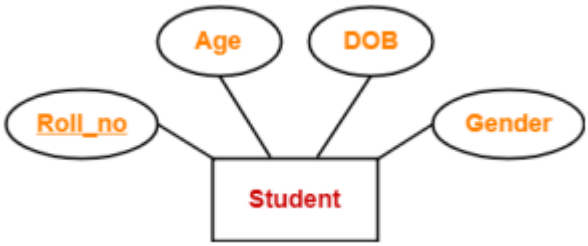
Composite attribute:

- An attribute which can be splitted into components is a composite attribute.
- Example: The address can be further splitted into house number, street number, city, state, country and pin code, the name can also be splitted into first name middle name and last name.



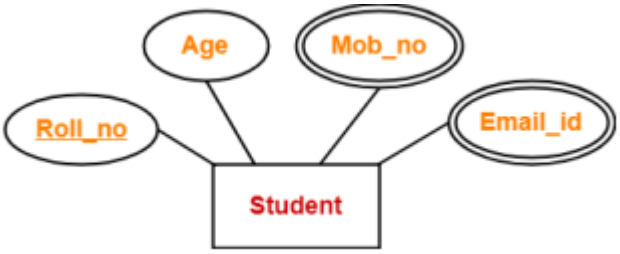
Single-valued attribute:

- The attribute which takes up only a single value for each entity instance is single-valued attribute.
- Example: The age of a student.



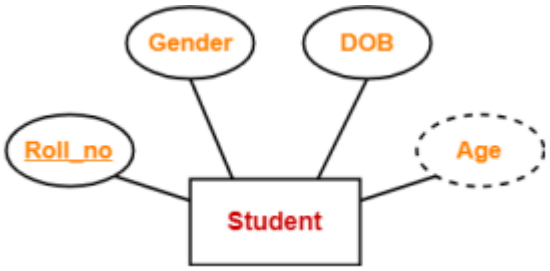
Multi-valued attribute:

- The attribute which takes up more than a single value for each entity instance is multi-valued attribute.
- Example: Phone number of a student: Landline and mobile.



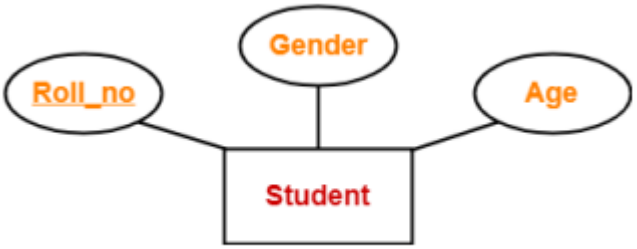
Derived attribute:

- An attribute that can be derived from other attributes is derived attribute.
- Example: Total and average marks of a student.



Key Attributes:

- Key attributes are those attributes which can identify an entity uniquely in an entity set.

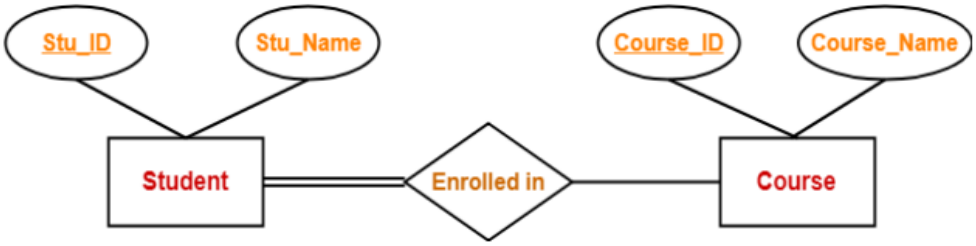


Entity set:

- An entity set is a set of entities of same type.
- An entity set may be of two types-
  - Strong Entity Set
  - Weak Entity Set.

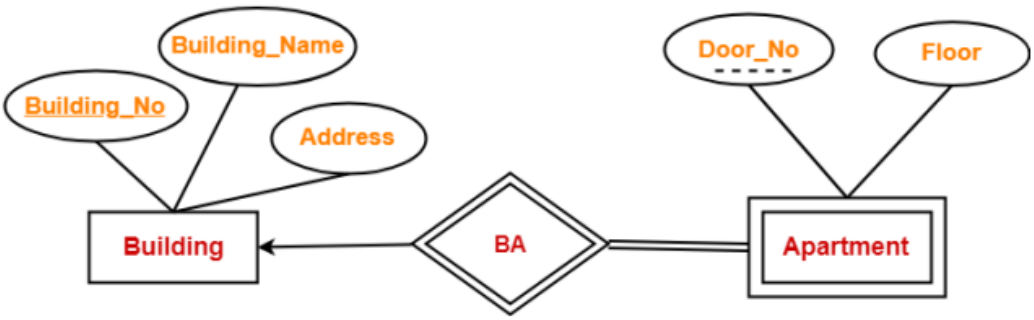
Strong Entity Set:

- A strong entity set is an entity set that contains sufficient attributes to uniquely identify all its entities.
- In other words, a primary key exists for a strong entity set.
- Primary key of a strong entity set is represented by underlining it.



Weak Entity Set:

- A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities.
- Weak entity depends on strong entity to ensure the existence of weak entity.
- In other words, a primary key does not exist for a weak entity set.
- However, it contains a partial key called as a discriminator.
- Discriminator can identify a group of entities from the entity set.
- Discriminator is represented by underlining with a dashed line.



RELATIONSHIPS AND RELATIONSHIP SETS:

- Relationship: A relationship is an association among two or more entities.
- The role of an entity is the function it plays in a relationship.
- For example, the relationship works- for could be ordered pairs of employee entities. The first employee takes the role of manager, and the second one will take the role of worker.
- A relationship may also have descriptive attributes. For example, date (last date of account access) could be an attribute of the CustAcct relationship set.
- A relationship set can be thought of as a set of n-tuples: Each n-tuple denotes a relationship involving n entities el through en, where entity ei is in entity set Ei.

Relationship set:

- A relationship set is a set of relationships of the same type.
- Degree of a Relationship Set: The number of entity sets that participate in a relationship set is termed as the degree of that relationship set. Thus,

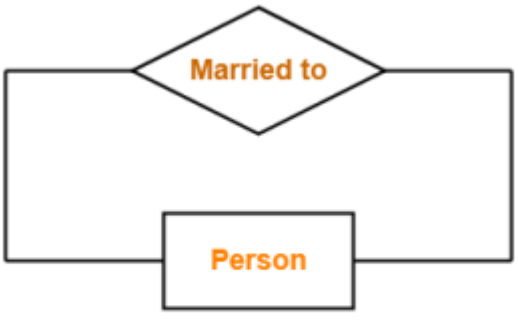
Types of Relationship Sets:

On the basis of degree of a relationship set, a relationship set can be classified into the following types-

1. Unary Relationship Set-

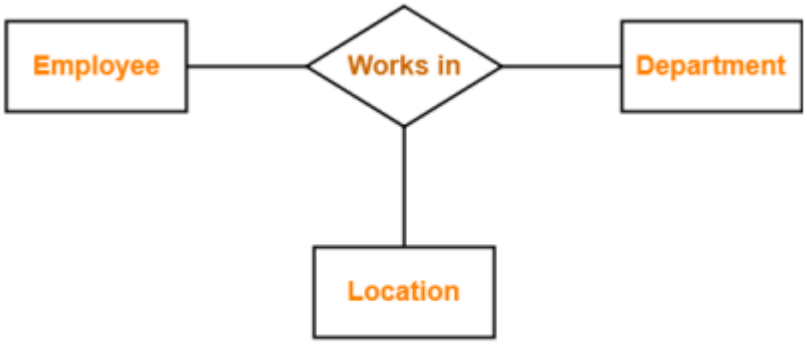
- Unary relationship set is a relationship set where only one entity set participates in a relationship set.

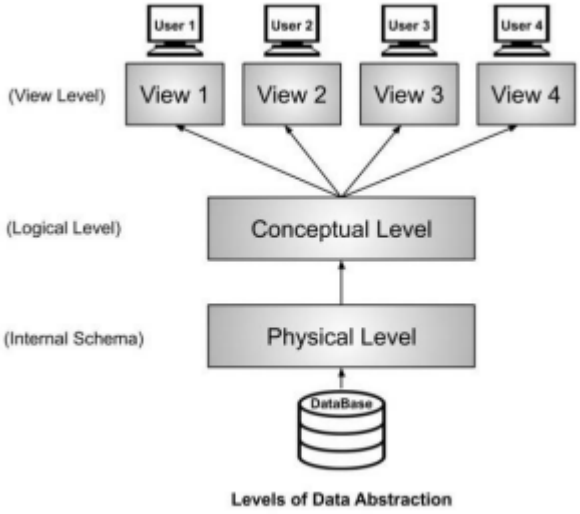
Example-



Unary Relationship Set

2. Binary Relationship Set:

	• Binary relationship set is a relationship set where two entity sets participate in a relationship set. Example:  Binary Relationship Set **3. Ternary Relationship Set:** • Ternary relationship set is a relationship set where three entity sets participate in a relationship set. Example:  Ternary Relationship Set	
2.	**What is database abstraction? Write in detail about levels of database abstraction?** • To store and retrieve data efficiently DBMS uses complex data structures. However as many users of DBMS are non computer professionals, DBMS hides the complexity and displays only the information that is required by the user. • The process of hiding complexities in accessing data is known as data abstraction. **Levels of Abstraction in a DBMS** • The data in a DBMS is described at three levels of abstraction. • The database description consists of a schema at each of these three levels of abstraction: Physical Schema (Internal) Conceptual Schema (Logical) External Schema (View)	2



Levels of Data Abstraction

Physical Schema (Internal):

- This is the layer which is containing the information about how the database is actually stored in the disk.
- The file structure of the database and the actual data constitute the physical level or physical view of the database.

Conceptual Schema(Logical):

- Conceptual level describes what data are actually stores in the database. It also describes the relationship existing among the data. In other words this level describes overall structural organization of the database.

External Schema (View):

- This level is closest to the user and is concerned with what data users want to view from the database.
- Most of the databases are not concerned with all the information contained in the database. Instead, they need only a part of the database relevant to them.
- The limited access to the database is known as view. Hence this layer is known as view layer.

Data Independence:

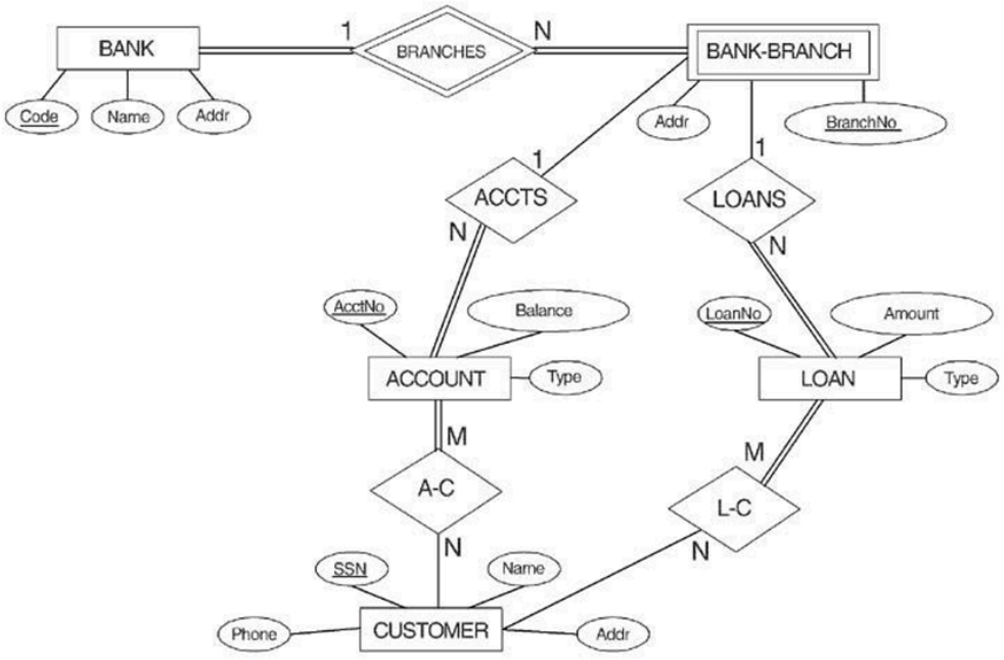
- A very important advantage of using a DBMS is that it offers data independence. That is, application programs are insulated from changes in the way the data is structured and stored.
- The separation of data from the programs that use the data is data independence. Nearly all modern applications are based on the principle of data independence.
- In fact, the whole concept of a database management system (DBMS) supports the notion of data independence since it represents a system for managing data separately from the programs that use the data.
- There are two levels of data independence:
    - Physical data independence
    - Logical data independence

Physical data independence:

- The ability to modify the physical scheme without causing application programs to be rewritten, when changes are made.
- Example: When one repartitions a table, add indexes or change a table's storage organization. Or in other words the old programs do not have to be rewritten, when changes are made to physical storage structure or physical devices on which data are stored.

	● Modifications at this level are usually to improve the performance. **Logical data independence:** ● The ability to modify the conceptual scheme without causing application programs to be rewritten. ● Example: We can add fields to the existing database or one can add views, triggers and constraints without affecting the application. We can delete the fields from the database if application programs do not use them. ● Logical data independence is harder to achieve as application programs are usually heavily dependent on logical structure of the data.	
3.	**List various categories of database users and discuss their interfaces to DBMS.** ● For a small personal database, such as storing the list of addresses one person typically defines, constructs, and manipulates the database ● However, in large organizations, many people are involved in the design, use, and maintenance of a large database with hundreds of users. They are called actors on the scene. ● Those who work to maintain the database system environment but who are not actively interested in the database contents are called workers behind the scene. **<u>Actors on the scene:</u>** **1. Database Administrators** ● In any database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software. ● Administering these resources is the responsibility of the database administrator (DBA) in an organization. ● The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed. ● The DBA is accountable for problems such as security breaches and poor system response time. 2. **Database Designers** ● Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. ● These tasks are mostly undertaken before the database is actually implemented and populated with data. ● The designer has to understand the requirements of the user and has to design the database. **3. End Users** ● End users are the people whose jobs require access to the database for querying, updating, and generating reports ● There are several categories of end users ● Casual end users occasionally access the database, but they may need different information each time. They use a database query language to specify their requests ● Naive or parametric end users make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates called	2

	canned transaction that have been carefully programmed and tested. ● Example: The tasks that such users perform are varied: Bank tellers check account balances and post withdrawals and deposits. Reservation agents for airlines, hotels, and car rental companies check availability for a given request and make reservations. ● Sophisticated end users include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements. ● Standalone users maintain personal databases by using program packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a tax package that stores a variety of personal financial data for tax purposes. **System Analysts and Application Programmers** (Software Engineers) ● System analysts determine the requirements of end users, especially naive and parametric end users, and ● They develop specifications for standard canned transactions that meet these requirements. ● Application programmers implement these specifications as program ● Then they test, debug, document, and maintain these canned transactions. ● Such analysts and programmers commonly referred to as software developers or software engineers	
4.	**List the various cases where use of a NULL value would be appropriate.** ● In SQL there may be some records in a table that do not have values or data for every field. ● This is possible because at a time of data entry some information is not available. ● So SQL supports a special value known as NULL which is used to represent the values of attributes that may be unknown or not apply to a tuple. ● SQL places a NULL value in the field in the absence of a user-defined value. ● For example, the Apartment_number attribute of an address applies only to address that are in apartment buildings and not to other types of residences. **Principles of NULL values:** ● Setting a NULL value is appropriate when the actual value is unknown, or when a value would not be meaningful. ● A NULL value is not equivalent to a value of ZERO if the data type is a number and is not equivalent to spaces if the data type is character. ● A NULL value can be inserted into columns of any data type. ● A NULL value will evaluate NULL in any expression. ● Suppose if any column has a NULL value, then UNIQUE, FOREIGN key, CHECK constraints will ignore by SQL. ● When a NULL is involved in a comparison operation, the result is considered to be UNKNOWN. Null value can be used when: 1. When the value of an attribute is irrelevant for an entity. For example: In a schema that stores information about a person if we have an attribute called Company, which stores the company name where a person works. If the person is unemployed we can store NULL value in the company attribute.	3

	2. If the entity does not have a value for an attribute. For example: If a database stores last_name, middle_name, first_name. If an entity has no middle_name we can place NULL. 3. When the value of a particular attribute is unknown. if the attribute Phone has a NULL value, it indicates that it is not known whether the value for Phone exists or not. ● We can use the IS NULL or IS NOT NULL operators to check for a NULL value.	
5.	**Consider the ER diagram shown in the figure below for part of a BANK database. Each bank can have multiple branches, and each branch can have multiple accounts and loans.**  	3

a. List the strong (non-weak) entity types in the ER diagram.

BANK, ACCOUNT, CUSTOMER, LOAN

b. Is there a weak entity type? If so, give its name, its partial key, and its identifying relationship.

Weak entity type: BANK-BRANCH.

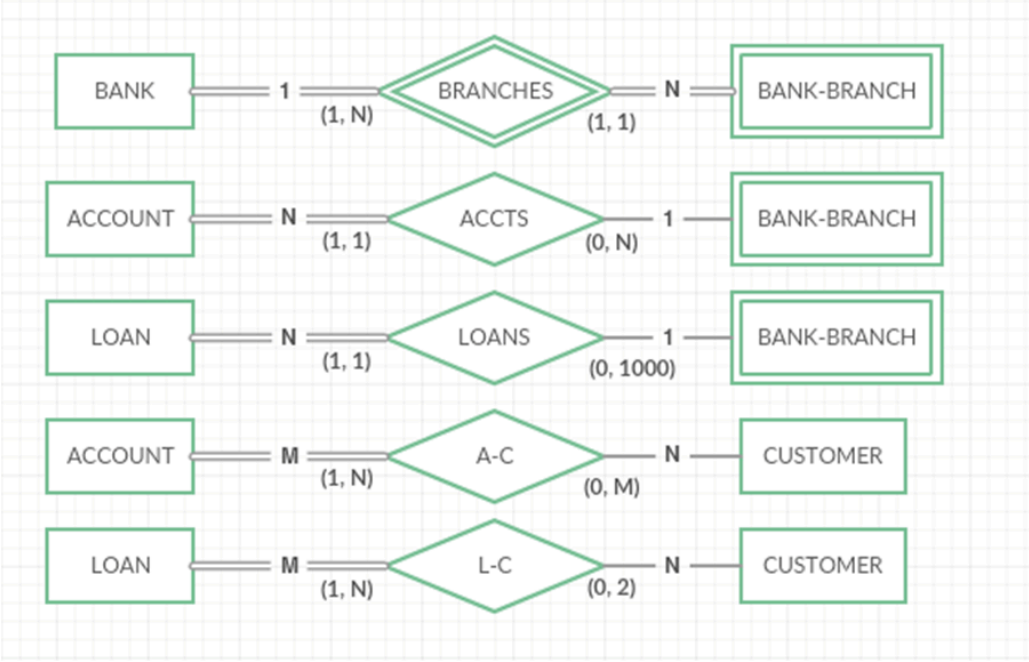
Partial key: BranchNo.

Identifying relationship: BRANCHES.

c. **What constraints do the partial key and the identifying relationship of the weak entity type specify in this diagram?**

- The partial key BranchNo in BANK-BRANCH specifies that the same BranchNo value may occur under different BANKs.
- The identifying relationship BRANCHES specifies thatBranchNo values are uniquely assigned for those BANK-BRANCH entities that are related to the same BANK entity.
- Hence, the combination of BANK Code and BranchNo togetherconstitute a full identifier for a BANK-BRANCH.

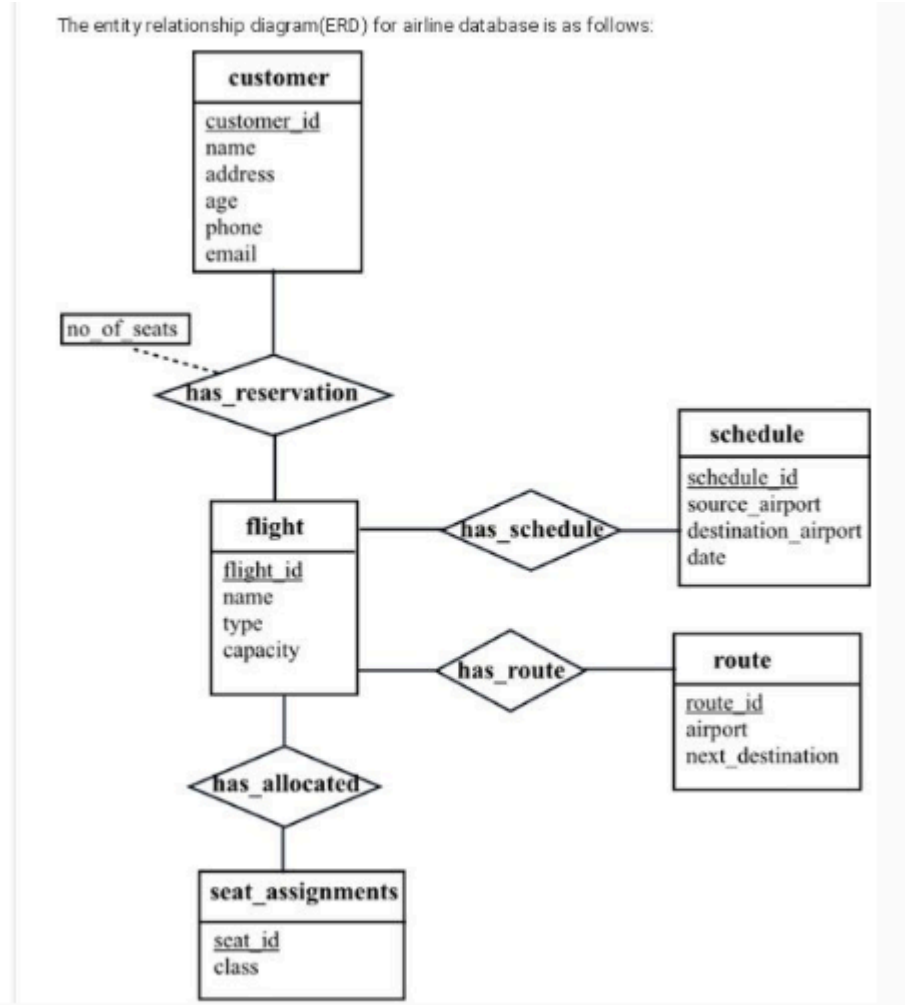
d. **List the names of all relationship types, and specify the (min,max) constraint on each participation of an entity type in a relationship type. Assume that a customer may have a maximum of 2 loans and that a maximum of 1000 loans are allowed per branch.**



	e. List concisely the user requirements that led to this ER schema design. ● Each BANK has a unique Code, as well as a Name and Address. ● Each BANK must haveone or more BANK-BRANCHes, and the BranhNo is unique among each set of BANK-BRANCHes that are related to the same BANK. ● Each BANK-BRANCH has an Address. ● Each BANK-BRANCH may have several LOANs and may have several ACCOUNTs. ● Each ACCOUNT has a unique AcctNo, Balance, and Type and must be related to exactly one BANK-BRANCH. ● Every ACCOUNT must be associated with at least one CUSTOMER. ● Each LOAN has a unique LoanNo, Amount, and Type and must be related to exactly one BANK-BRANCH ● Every LOAN must be associated with at least one CUSTOMER. ● Each CUSTOMER has a unique SSN, Name, Phone, and Address, and may have several ACCOUNTs and may have several LOANs.	
6.	**Ram wants to design a Web-based system to make airline reservations and sell airline tickets, which DBMS architecture would you choose? Why? Why would the other architectures not be a good choice?** **a. The best architecture** The best architecture for a web design will be a Three-tier client/server architecture because: 1. The web user interface is placed in the client system. 2. The database server contains the DBMS 3. Web/application server will handle the application logic of the system 4. Web server can handle those transactions validate the data and manipulate database accordingly. 5. The web user interface is vital as different types of users e.g. naive users interact with the system **b. Architectures That are not suitable** 1. In a centralized DBMS architecture, DBMS functionality and user interface are both performed on the same system hence not appropriate for web based system. 2. In a Tier-Tier client/Server architecture can be a burden if business logic is placed in database server. **b) Design a database for an airline. The database must keep track of customers and their reservations, flights and their status, seat assignments on individual flights, and the schedule and routing of future flights. Your design should include an E-R diagram, a set of relational schemas, and a list of constraints,**	3

including primary-key and foreign-key constraints.

Airline Database Design



Relation shema :

The relation schemas are as follows:

customer(customer_ID, name, address, age, phone, email)

flight(flight_id, name, type, capacity)

schedule(schedule_id, source_airport, destination_airport,

date)

route(route_id, airport, next_destination)

seat_assignements(seat_id, class)

has_reservation(customer_id, flight_id, no_of_seats)

has_allocated(flight_id, customer_ID, seat_id)

has_schedule(flight_id, schedule_id)

has_route(flight_id, route_id)

Primary Keys of the Schema:

The primary keys of the relation schemas are as follows:

- customer with the primary key customer_ID.
- flight with the primary key flight_ID.
- Schedule with the primary key schedule_ID.
- route with the primary key route_ID.
- seat_assignements with the primary key seat_ID.
- has_reservation with the composite primary key customer_ID, flight_ID.
- has_allocated with the composite primary key customer_ID, flight_ID.
- has_schedule with the composite primary key flight_ID, schedule_ID.
- has_route with the composite primary key flight_ID, route_ID.

ForeignKeys of the schema:

	A foreign key is an attribute/column which is used for establishing relationship between two tables/relations. The foreign keys of the relation schemas are as follows: • In has_reservation relation, customer_id and flight_id are foreign keys. o customer_id references customer_id in customer relation. o flight_id references flight_ID in flight relation. • In has_allocated relation, customer_id and flight_id are foreign keys. o customer_id references customer_id in customer relation. o flight_id references flight_ID in flight relation. • In has_schedule relation, flight_id and schedule_id are foreign keys. o flight_id references flight_ID in flight relation. o schedule_id references schedule_id in schedule relation. • In has_route relation, flight_id and route_id are foreign keys. o flight_id references flight_ID in flight relation. o route_id references route_id in route relation.	
7.	**Discuss the main categories of data models. What are the basic differences between the relational model, E-r model.** **E-R model:** Refer Qno1 **Relational model:** Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute. **Domain:** It contains a set of atomic values that an attribute can take. **Attribute:** It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom(Ai) **Relational instance:** In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples. **Relational schema:** A relational schema contains the name of the relation and name of all columns or attributes. **Relational key:** In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely. **Example: STUDENT Relation**	4

NAME	ROLL_NO	PHONE_NO	ADDRESS	AGE
Ram	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20
Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

- In the given table, NAME, ROLL_NO, PHONE_NO, ADDRESS, and AGE are the attributes.
- The instance of schema STUDENT has 5 tuples.
- t3 = <Laxman, 33289, 8583287182, Gurugram, 20>

Properties of Relations

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Attribute domain has no significance
- tuple has no duplicate value
- Order of tuple can have a different sequence

Differences between E-R model and relational model:

- The main distinction between the ER model and the Relational Model is that the ER model describes the relationship between entities and their attributes. On the other hand, the Relational Model referred to the implementation of our model.
- The Relational Model is the implementation or representational model, while the ER Model is the high-level or conceptual model.
- The data in components such as entity sets, relationship sets, and attributes are represented by an ER model. The Relational model, on the other hand, defines data in components such as tuples, attributes, and attribute domains.
- As compared to a Relational Model, an ER model makes it easier to understand the relationships between entities.
- Mapping Cardinality is always a constraint in the ER model, while the cardinality constraint cannot be defined in the Relational Model.

8. **Differentiate database system versus files system.** 4

S.NO.	File System	DBMS
1.	File system is software that manages and organizes the files in a storage medium within a computer.	DBMS is software for managing the database.
2.	Redundant data can be present in a file system.	In DBMS there is no redundant data.
3.	It doesn't provide backup and recovery of data if it is lost.	It provides backup and recovery of data even if it is lost.
4.	There is no efficient query processing in file system.	Efficient query processing is there in DBMS.
5.	There is less data consistency in file system.	There is more data consistency because of the process of normalization.
6.	It is less complex as compared to DBMS.	It has more complexity in handling as compared to file system.
7.	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file system.
8.	It is less expensive than DBMS.	It has a comparatively higher cost than a file system.

Advantages of a DBMS over File system:
Using a DBMS to manage data has many advantages:

Data Independence:
- Application programs should be as independent as possible from details of data representation and storage.
- The DBMS can provide an abstract view of the data to insulate application code from such details.

Efficient Data Access:
- A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently.
- This feature is especially important if the data is stored on external storage devices.

Data Integrity and Security:
- If data is always accessed through the DBMS, the DBMS can enforce integrity constraints on the data.
- For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded.
- Also, the DBMS can enforce access controls that govern what data is visible to different classes of users.

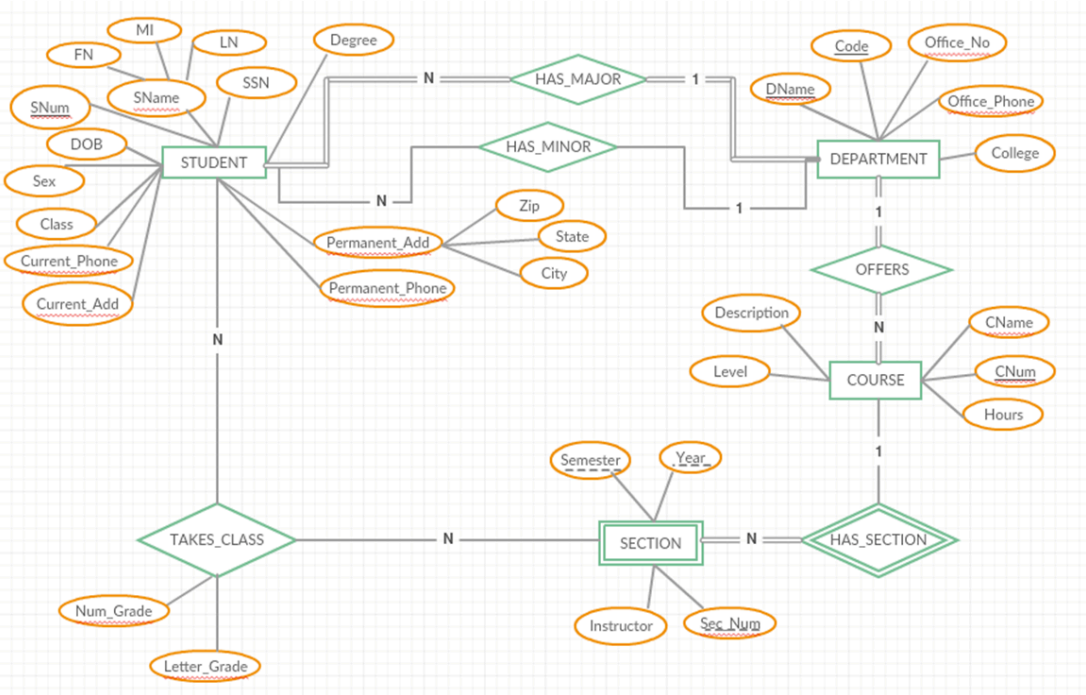
Concurrent Access and Crash Recovery:
- A database system allows several users to access the database concurrently.
- The concurrent use of data increases the economy of a system.

	● A DBMS also protects data from failures such as power failures and crashes etc. by the recovery schemes such as backup mechanisms and log files etc. **Data administration:** ● When several users share the data, centralizing the administration of data can offer significant improvements. ● Organization of data minimizes redundancy and fine-tuning the storage of the data to make efficient retrieval. **Reduced Application Development Time:** ● DBMS supports many important functions that are common to many applications accessing data stored in the DBMS.	
9.	**Explain the architecture of DBMS with a neat sketch.**	5
10.	**Consider the following set of requirements for a UNIVERSITY database that is used to keep track of students' transcripts.** **a.The university keeps track of each student's name, student number, social security number, current address and phone, permanent address and phone, birthdate, sex, class (freshman, sophomore, ..., graduate), major department, minor department (if any), and degree program (B.A., B.S., ..., Ph.D.). Some user applications need to refer to the city, state, and zip of the student's permanent address, and to the student's last name. Both social security number and student number have unique values for each student.** **b. Each department is described by a name, department code, office number, office phone, and college. Both name and code have unique values for each department.** **c. Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of course number is unique for each course.** **d. Each section has an instructor, semester, year, course, and section number. The section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number**	6

of sections taught during each semester.

e. A grade report has a student, section, letter grade, and numeric grade (0, 1, 2, 3,4 for F, D, C, B, A, respectively).

Design an ER schema for this application, and draw an ER diagram for that schema.Specify key attributes of each entity type and structural constraints on each relationship type. Note any unspecified requirements, and make appropriate assumptions to make the specification complete.



Assumptions

- Every department must have at least one student enrolled in it as a major
- A student must specify his major when he/she joins the university
- Not all departments are used as a minor department for a student's degree
- Every department must offer at least one course
- Every course must belong to exactly one departme
- Not all courses will have sections (a course may be new and will not be offered until next year, or there is no instructor available to teach the course
- A section may not have any students enrolled in it
- A student may not sign up for any classes (sections) (for example, the student dropped that semester for medical reasons

QNo	Question	BTL

1.	**Define canned transactions.** Canned transactions are standard types of queries and updates which frequently used by Naive or parametric end users to constantly querying and updating database. ● Example: The tasks that such users perform are varied: Bank tellers check account balances and post withdrawals and deposits. Reservation agents for airlines, hotels, and car rental companies check availability for a given request and make reservations.	1
2.	**What are the four types of actions which involve databases. Briefly discuss each.** The DBMS is a software system that explains the four types of actions, which are defining, constructing, manipulating, and sharing databases among various users and applications. 1. **Defining a database**: It includes the data types, structures, and constraints of the data have to store in the database. The database descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called meta-data. 2. **Constructing the database**: It is the process of data storing on some storage medium that is maintained by the DBMS. 3. **Manipulating a database**: It includes functions such as retrieve the database by using query, updating the database to reflect changes in the system, and generate reports from the data. 4. **Sharing a database**: It allows multiple users and programs to access the database simultaneously.	2
3.	**List some integrity constraints that can apply to the database.**	3

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

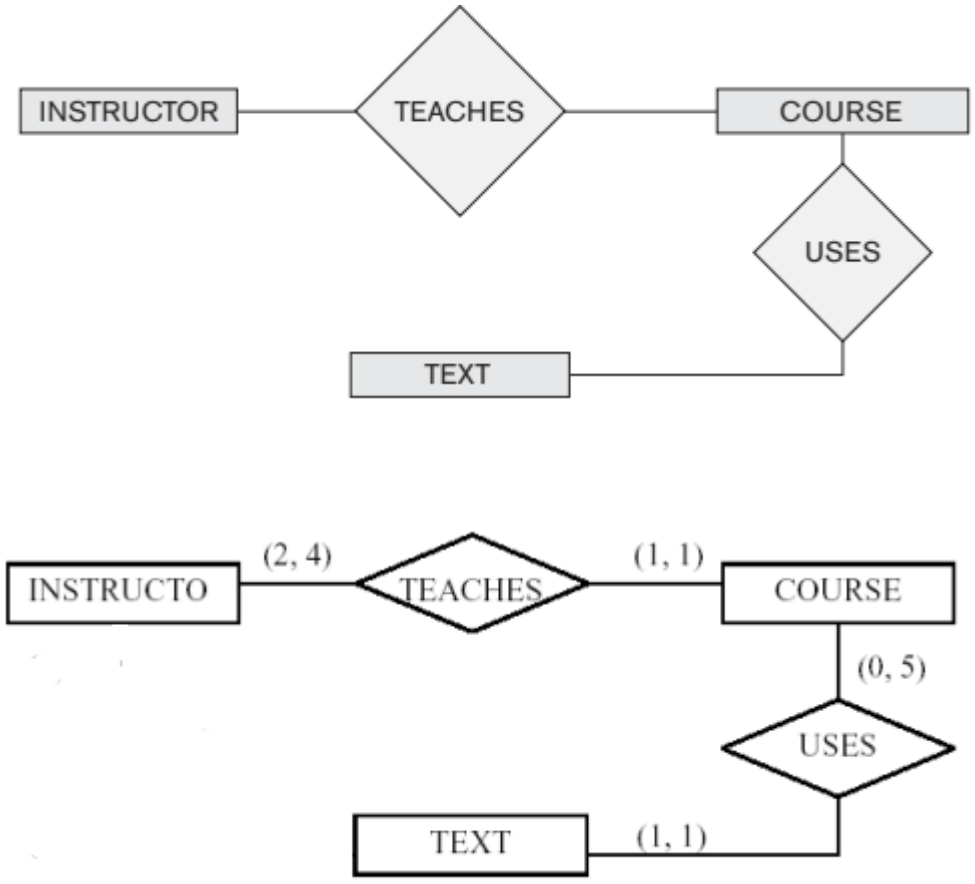
Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

(a) The StudentNumber should be unique for each STUDENT record (key constraint).
(b) The CourseNumber should be unique for each COURSE record (key constraint).
(c) A value of CourseNumber in a SECTION record must also exist in some COURSE record (referential integrity constraint).
(d) Every record in COURSE must have a value for CourseNumber (entity integrity constraint).

4. **Discuss the capabilities that can be provided by DBMS.** — 2

- Controlling Redundancy: normalization
- Restricting unauthorized Access
- Multiple User interfaces: query language, programming language interfaces (forms and command codes)
- Representing Complex Relationships among Data
- Enforcing Integrity Constraints
- Providing Persistent storage for program objects
- Providing storage structures for efficient query processing
- Backup and recovery

5. **Consider the below ER diagram. Assume that a course may or may not use a textbook, but that a text by definition is a book that is used in some course. A course may not use more than five books. Instructors teach from two to four courses. Supply (min, max) constraints on this diagram.** — 3

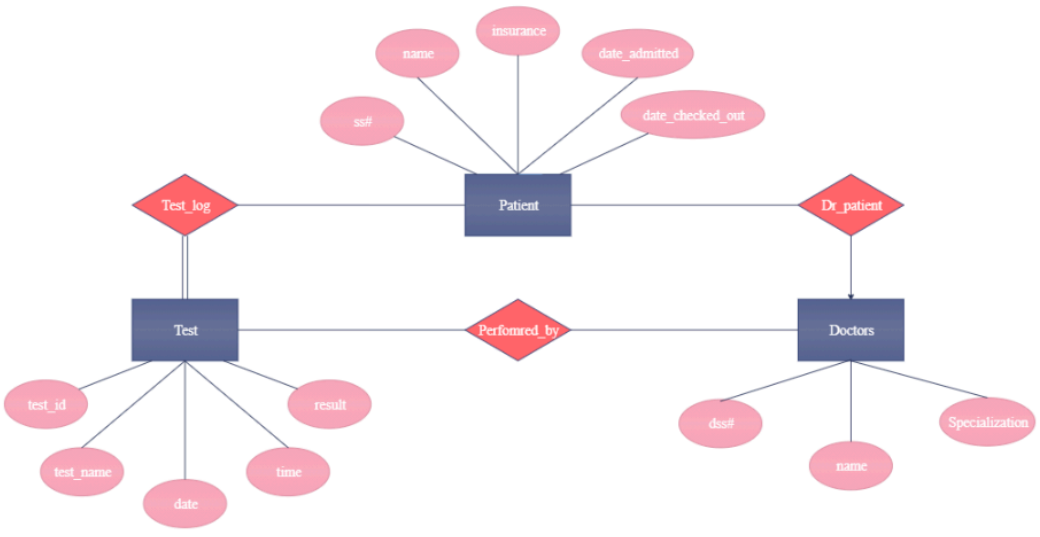


6. **Differentiate between a database schema and a database state?** — 4

- Database Schema is the overall Design of the Database. It is the skeleton structure that represents the logical view of the entire database. It tells how the data is organized and how the relations among them are associated.
- Database State: Refers to the content of a database at a moment in time.

7. **Differentiate between database and DBMS.** — 4

Database	DBMS
A database is a collection of connected information about people, locations, or things.	A database management system (DBMS) is a collection of programs that allow you to create, manage, and operate a database.
Data is maintained in physical ledgers, books, or papers.	All the records are maintained only on a computer.
The retrieval of information from the databases can be done manually, through queries, or by using programs (C, C++, Java, etc.).	We can retrieve the data from the database management system through queries written in SQL.
Data retrieval is slow.	Data retrieval is very quick.

	The databases are not designed for a large number of people who can access data at the same time.	The database management system is designed for a large number of people who can access the data at the same time.
	Very less information can be modified at a time.	A lot of information can be changed at one time
	The databases do not ensure that the data will be available after failure arises.	The database management system (DBMS) ensures that the data will always be available even after system failures
8.	**Specify the relationship among the following relations** **1. Student(Name, Stu_Num,Class, Major)** **2. Course(Course_name, Course_num, Credit_hours, Dept)** **3. Section(Sec_id,Course_num,Sem,Year,Instructor)** **4. Grade(Stu_Num,Sec_id, Grade)** **5. Prerequisite(Course_num, Pre_id)** (a) Each SECTION record is related to a COURSE record. (b) Each GRADE REPORT _REPORT record is related to one STUDENT record and one SECTION record. (c) Each PREREQUISITE record relates two COURSE records: one in the role of a course and the other in the role of a prerequisite to that course.	3
9.	**Give examples of systems in which it may make sense to use traditional file processing instead of a database approach.** 1. Small internal utility to locate files 2. Small single user application that does not require security (such as a customized calculator or a personal address and phone book) 3. Real-time navigation system (with heavy computation and very little data)	5
10.	**Design an E-R model for Hospital Management System.** 	6

QNo	Question	BTL
1.	**Describe the four clauses in the syntax of a simple SQL retrieval query.** ● Select: list of attribute names to be received by the query ● From: the table ● s that these attributes with be retrieved from ● Where: conditional boolean expression to identify certain tuples to be retrieved (optional) ● Order by: attribute list to order the result by (optional)	2
2.	**List the data types that are allowed for SQL attributes.** ● Numeric: include integer numbers of various sizes and floating-point (real) numbers of various precision ● Character- string: fixed length characters. ● Bit-string: fixed length n—BIT(n)—or varying length—BIT VARYING(n), where n is the maximum number of bits. ● Boolean: Traditional values true or false ● Date: components are YEAR, MONTH, and DAY in the form YYYY-MM-DD. ● Time: components are YEAR, MONTH, and DAY in the form YYYY-MM-DD.	2
3.	**Discuss how NULLs are treated in comparison operators in SQL.** If an aggregate function against a column that contains nulls is executed, the function ignores the nulls. This prevents unknown or inapplicable values from affecting the result of the aggregate.	2
4.	**Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:** **1. STUDENT (SSN, Name, Major, Bdate)** **2. COURSE (Course#, Quarter, Grade)** **3. ENROLL (SSN, Course#, Quarter, Grade)** **4. BOOK_ADOPTION (Course#, Quarter, Book_ISBN)** **5. TEXT (Book_ISBN, Book_Title, Publisher, Author)** **Specify the foreign keys for the schema.** The schema of this question has the following four foreign keys: a. the attribute SSN of relation ENROLL that references relation STUDENT, b. the attribute Course# in relation ENROLL that references relation COURSE, c. the attribute Course# in relation BOOK_ADOPTION that references relation COURSE, and d. the attribute Book_ISBN of relation BOOK_ADOPTION that references relation TEXT..	3
5.	**Suggest a query to calculate the average salary of all employees in employee relation on salary attribute.** Avg aggregate function is used to calculate the average of values regarding to an attribute. The query to calculate the average of salary in an employee relation is	3

	Select avg(salary) as Average from employee;	
6.	**How are inner join operation differ from outer join.** **Inner Join :** It is a type of join operation in SQL. Inner join is an operation that returns combined tuples between two or more tables where at least one attribute is in common. If there is no attribute in common between tables then it will return nothing. **Outer Join:** It is a type of Join operation in SQL. Outer join is an operation that returns combined tuples from a specified table even if the join condition fails.	3
7.	**When a query language called relationally complete.** ● If a language is relationally complete, it means that queries of arbitrary complexity can be formulated without having to resort to iterative loops or branching. ● In other words, it is relational completeness that allows end users to access the database directly, without having to implement repetitive constructs such as 1)for, 2) while or 3) do-while loops.	3
8.	**Differentiate between delete and truncate command.** 1. The DELETE statement is used when we want to remove some or all of the records from the table, while the TRUNCATE statement will delete entire rows from a table. 2. DELETE is a DML command as it only modifies the table data, whereas the TRUNCATE is a DDL command. 3. DELETE command can filter the record/tuples by using the WHERE clause. However, the TRUNCATE command does not allow to use **WHERE** clause, so we cannot filter rows while truncating. 4. DELETE statement only deletes records and does not reset the **table's identity**, whereas TRUNCATE resets the identity of a particular table.	4
9.	**How to drop the primary key in MySQL?** MySQL primary key is a single or combination of the field used to identify each record in a table uniquely. A primary key column cannot be null or empty. We can remove or delete a primary key from the table using the ALTER TABLE statement. The following syntax is used to drop the primary key: 1. **ALTER TABLE** table_name **DROP PRIMARY KEY**;	1
10.	**Retrieve the minimum number of working hours of employees from employee table with attributes emp_name, emp_id, working_hours.** SELECT emp_name, MIN(working_hours) AS "Minimum working hour" FROM	6

	employees GROUP BY emp_name;	

QNo	Question	BTL
1.	**Discuss about the different constraints used in SQL.** ● The constraint in MySQL is used to specify the rule that allows or restricts what values/data will be stored in the table ● Constraints in MySQL is classified into two types: ● Column Level Constraints: These constraints are applied only to the single column that limits the type of particular column data. ● Table Level Constraints: These constraints are applied to the entire table that limits the type of data for the whole table. ● We can define the constraints during a table created by using the CREATE TABLE statement. **Syntax** **CREATE TABLE** new_table_name (col_name1 datatype **constraint**, col_name2 datatype **constraint**, col_name3 datatype **constraint**, ...); **Constraints used in mysql** ● NOT NULL ● CHECK ● DEFAULT ● PRIMARY KEY ● AUTO_INCREMENT ● UNIQUE **NOT NULL** This constraint specifies that the column cannot have NULL or empty values.	2

```
mysql> create table student(name varchar(20) not null, rollno int, phone varchar(10));
Query OK, 0 rows affected (2.30 sec)

mysql> insert into student values(null,1,1234566890);
ERROR 1048 (23000): Column 'name' cannot be null
```

```
mysql> desc student;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| name   | varchar(20) | NO   |     | NULL    |       |
| rollno | int         | YES  |     | NULL    |       |
| phone  | varchar(10) | YES  |     | NULL    |       |
| marks  | int         | NO   |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

UNIQUE

This constraint ensures that all values inserted into the column will be unique. It means a column cannot stores duplicate values.

```
mysql> alter table student modify column rollno int unique;
Query OK, 0 rows affected (1.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| name   | varchar(20) | NO   |     | NULL    |       |
| rollno | int         | YES  | UNI | NULL    |       |
| phone  | varchar(10) | YES  |     | NULL    |       |
| marks  | int         | NO   |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> INSERT INTO STUDENT VALUES("RAM",1,9876543210,24);
ERROR 1146 (42S02): Table 'dbpavani.STUDENT' doesn't exist
mysql> INSERT INTO student VALUES("RAM",1,9876543210,24);
Query OK, 1 row affected (0.30 sec)

mysql> select * from student;
+------+--------+------------+-------+
| name | rollno | phone      | marks |
+------+--------+------------+-------+
| RAM  |      1 | 9876543210 |    24 |
+------+--------+------------+-------+
1 row in set (0.00 sec)

mysql> INSERT INTO student VALUES("raj",1,9876543210,24);
ERROR 1062 (23000): Duplicate entry '1' for key 'student.rollno'
```

CHECK:

It controls the value in a particular column. It ensures that the inserted value in a column must be satisfied with the given condition.

CHECK (expr)

```
mysql> alter table student ADD COLUMN age int check(age>18);
Query OK, 1 row affected (4.06 sec)
Records: 1  Duplicates: 0  Warnings: 0


mysql> insert into student values("sita",2,8765943221,23,15);
ERROR 3819 (HY000): Check constraint 'student_chk_1' is violated.
mysql> desc student;
```

DEFAULT:

This constraint is used to set the default value for the particular column where we have not specified any value.

```
mysql> alter table student MODIFY COLUMN age int DEFAULT 20;
Query OK, 0 rows affected (0.38 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> insert into student(name,rollno,phone,marks) values("sita",8,8765943221,25);
Query OK, 1 row affected (0.21 sec)

mysql> select * from student;
+------+--------+------------+-------+------+
| name | rollno | phone      | marks | age  |
+------+--------+------------+-------+------+
| RAM  |      1 | 9876543210 |    24 | NULL |
| sita |      5 | 8765943221 |    23 | NULL |
| sita |      6 | 8765943221 |    23 | NULL |
| sita |      8 | 8765943221 |    25 |   20 |
+------+--------+------------+-------+------+
4 rows in set (0.00 sec)
```

PRIMARY KEY:

- This constraint is used to identify each record in a table uniquely.
- If the column contains primary key constraints, then it cannot be null or empty.
- It can contain only one column as primary key. It always contains unique value into a column.

```
mysql> alter table student modify COLUMN rollno int primary key;
Query OK, 0 rows affected (3.51 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| name   | varchar(20) | NO   |     | NULL    |       |
| rollno | int         | NO   | PRI | NULL    |       |
| phone  | varchar(10) | YES  |     | NULL    |       |
| marks  | int         | NO   |     | NULL    |       |
| age    | int         | YES  |     | 20      |       |
+--------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> select * from student;
+------+--------+------------+-------+------+
| name | rollno | phone      | marks | age  |
+------+--------+------------+-------+------+
| RAM  |      1 | 9876543210 |    24 | NULL |
| sita |      5 | 8765943221 |    23 | NULL |
| sita |      6 | 8765943221 |    23 | NULL |
| sita |      8 | 8765943221 |    25 |   20 |
+------+--------+------------+-------+------+
4 rows in set (0.01 sec)
```

```
mysql> select * from student;
+------+--------+------------+-------+------+
| name | rollno | phone      | marks | age  |
+------+--------+------------+-------+------+
| RAM  |      1 | 9876543210 |    24 | NULL |
| sita |      5 | 8765943221 |    23 | NULL |
| sita |      6 | 8765943221 |    23 | NULL |
| sita |      8 | 8765943221 |    25 |   20 |
+------+--------+------------+-------+------+
4 rows in set (0.01 sec)

mysql> insert into student values("ravi",6,8765943221,23,NULL);
ERROR 1062 (23000): Duplicate entry '6' for key 'student.PRIMARY'
mysql> insert into student(name,phone,marks,age) values("ravi",8765943221,23,NULL);
ERROR 1364 (HY000): Field 'rollno' doesn't have a default value
mysql> insert into student(name,rollno,phone,marks,age) values("ravi",null,8765943221,23,NULL);
ERROR 1048 (23000): Column 'rollno' cannot be null
```

AUTO INCREMENT:

This constraint automatically generates a unique number whenever we insert a new record into the table.The constraint should be used with key.

```
mysql> create table store(id int not null auto_increment primary key,name varchar(10));
Query OK, 0 rows affected (2.42 sec)

mysql> insert into store(name) values("soap"),("milk"),("pen");
Query OK, 3 rows affected (0.20 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from store;
+----+------+
| id | name |
+----+------+
|  1 | soap |
|  2 | milk |
|  3 | pen  |
+----+------+
3 rows in set (0.00 sec)
```

2. **Discuss about triggers.** 2

A trigger is a special type of stored procedure that is invoked automatically in response to an event. Each trigger is associated with a table, which is activated on any DML statement such as INSERT, UPDATE, or DELETE.

Types of Triggers in MySQL

We can define the maximum six types of actions or events in the form of triggers:

1. **Before Insert:** It is activated before the insertion of data into the table.
2. **After Insert:** It is activated after the insertion of data into the table.
3. **Before Update:** It is activated before the update of data in the table.
4. **After Update:** It is activated after the update of the data in the table.
5. **Before Delete:** It is activated before the data is removed from the table.
6. **After Delete:** It is activated after the deletion of data from the table.

- We can create a trigger using

CREATE TRIGGER trigger_name
(**AFTER** | BEFORE) (**INSERT** | **UPDATE** | **DELETE**)
ON table_name **FOR** EACH ROW
BEGIN
--variable declarations
--trigger code
END;

After Insert Trigger

After Insert Trigger in MySQL is invoked automatically whenever an insert event occurs on the table.

CREATE TRIGGER trigger_name

AFTER INSERT

ON table_name **FOR** EACH ROW

trigger_body ;

Example:

```
mysql>  Create Trigger after_insert_details
    -> AFTER INSERT ON student_info FOR EACH ROW
    -> BEGIN
    -> INSERT INTO student_detail VALUES (new.stud_id, new.stud_code,
    -> new.stud_name, new.subject, new.marks, new.phone, CURTIME());
    -> END //
Query OK, 0 rows affected (0.12 sec)
```

```
mysql> INSERT INTO student_info VALUES
    -> (10, 110, 'Alexandar', 'Biology', 67, '2347346438');
Query OK, 1 row affected (0.09 sec)

mysql> SELECT * FROM student_detail;
+---------+-----------+-----------+---------+-------+------------+-------------+
| stud_id | stud_code | stud_name | subject | marks | phone      | Lasinserted |
+---------+-----------+-----------+---------+-------+------------+-------------+
|      10 | 110       | Alexandar | Biology |    67 | 2347346438 | 14:41:35    |
+---------+-----------+-----------+---------+-------+------------+-------------+
1 row in set (0.00 sec)
```

In this output on inserting values into the student_info table, the student_detail table will automatically fill the records by invoking a trigger.

Before Insert Trigger

Before Insert Trigger in MySQL is invoked automatically whenever an insert operation is executed

CREATE TRIGGER trigger_name

BEFORE **INSERT**

ON table_name **FOR** EACH ROW

Trigger_body ;

Example:

```
CREATE TABLE employee(
   name varchar(45) NOT NULL,
   occupation varchar(35) NOT NULL,
   working_date date,
   working_hours varchar(10)
);
```

we will use a CREATE TRIGGER statement to create a BEFORE INSERT trigger. This trigger is invoked automatically that inserts the occupation = 'Leader' if someone tries to insert the occupation = 'Scientist'.

```
mysql> DELIMITER //
mysql> CREATE TRIGGER before_insert_occupation
    -> BEFORE INSERT ON employee FOR EACH ROW
    -> BEGIN
    -> IF NEW.occupation = 'Scientist' THEN SET NEW.occupation = 'Doctor';
    -> END IF;
    -> END //
Query OK, 0 rows affected (0.18 sec)
```

```
mysql> INSERT INTO employee VALUES
    -> ('Markus', 'Scientist', '2020-10-08', 14);
Query OK, 1 row affected (0.13 sec)

mysql> INSERT INTO employee VALUES
    -> ('Alexander', 'Actor', '2020-10-012', 13);
```

After the execution of above statement the output will be

name	occupation	working_date	working_hours
Robin	Scientist	2020-10-04	12
Warner	Engineer	2020-10-04	10
Peter	Actor	2020-10-04	13
Marco	Doctor	2020-10-04	14
Brayden	Teacher	2020-10-04	12
Antonio	Business	2020-10-04	11
Markus	Doctor	2020-10-08	14
Alexander	Actor	2020-10-12	13

BEFORE UPDATE Trigger:

BEFORE UPDATE Trigger in MySQL is invoked automatically whenever an update

operation is fired on the table associated with the trigger.

CREATE TRIGGER trigger_name

BEFORE **UPDATE**

ON table_name **FOR** EACH ROW

trigger_body ;

Example:

```
mysql>
mysql> DELIMITER $$
mysql> CREATE TRIGGER before_update_salesInfo
    -> BEFORE UPDATE
    -> ON sales_info FOR EACH ROW
    -> BEGIN
    ->     DECLARE error_msg VARCHAR(255);
    ->     SET error_msg = ('The new quantity cannot be greater than 2 times the current quantity');
    ->     IF new.quantity > old.quantity * 2 THEN
    -> SIGNAL SQLSTATE '45000'
    -> SET MESSAGE_TEXT = error_msg;
    ->     END IF;
    -> END $$
Query OK, 0 rows affected (0.26 sec)
```

```
mysql> DELIMITER ;
mysql> UPDATE sales_info SET quantity = 125 WHERE id = 2;
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE sales_info SET quantity = 600 WHERE id = 2;
ERROR 1644 (45000): The new quantity cannot be greater than 2 times the current quantity
```

AFTER UPDATE Trigger:

The AFTER UPDATE trigger in MySQL is invoked automatically whenever an UPDATE event is fired on the table associated with the triggers.

CREATE TRIGGER trigger_name

AFTER UPDATE

ON table_name **FOR** EACH ROW

trigger_body ;

Example:

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER after_update_studentsInfo
    -> AFTER UPDATE
    -> ON students FOR EACH ROW
    -> BEGIN
    ->     INSERT into students_log VALUES (user(),
    -> CONCAT('Update Student Record ', OLD.name, ' Previous Class :',
    -> OLD.class, ' Present Class ', NEW.class));
    -> END $$
Query OK, 0 rows affected (0.23 sec)
```

```
mysql> UPDATE students SET class = class + 1;
Query OK, 4 rows affected (0.15 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> SELECT * FROM students;
+----+-----------+-------+-------------------------+
| id | name      | class | email_id                |
+----+-----------+-------+-------------------------+
|  1 | Stephen   |     7 | stephen@javatpoint.com  |
|  2 | Bob       |     8 | bob@javatpoint.com      |
|  3 | Steven    |     9 | steven@javatpoint.com   |
|  4 | Alexandar |     8 | alexandar@javatpoint.com |
+----+-----------+-------+-------------------------+
```

3. **Explain about DDL commands.** 1

Create command

MySQL allows us to create a table into the database by using the **CREATE TABLE** command. Following is a generic **syntax** for creating a MySQL table in the database.

```
CREATE TABLE [IF NOT EXISTS] table_name(
   column_definition1,
   column_definition2,
   ........,
   table_constraints
);
```

The parameter descriptions of the above syntax are as follows:

- Database_name: It is the name of a new table. It should be unique in the MySQL database that we have selected. The **IF NOT EXIST** clause avoids an error when we create a table into the selected database that already exists.
- Column_definition: It specifies the name of the column along with data types for each column. The columns in table definition are separated by the comma operator.
- Table_constraints: It specifies the table constraints such as PRIMARY KEY, UNIQUE KEY, FOREIGN KEY, CHECK, etc.

Example:

```
mysql> CREATE TABLE employee_table(
    -> id int NOT NULL AUTO_INCREMENT,
    -> name varchar(45) NOT NULL,
    -> occupation varchar(35) NOT NULL,
    -> age int NOT NULL,
    -> PRIMARY KEY (id)
    -> );
Query OK, 0 rows affected (1.47 sec)
```

- We can use desc or describe command to describe the structure of the table.

```
mysql> DESCRIBE employee_table;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int         | NO   | PRI | NULL    | auto_increment |
| name       | varchar(45) | NO   |     | NULL    |                |
| occupation | varchar(35) | NO   |     | NULL    |                |
| age        | int         | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

MySQL ALTER Table

MySQL ALTER statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.

The ALTER statement is always used with "ADD", "DROP" and "MODIFY" commands according to the situation.

1) **ADD a column in the table**

Syntax:

ALTER TABLE table_name
ADD new_column_name column_definition
[**FIRST** | **AFTER** column_name];

Example:

In this example, we add a new column "cus_age" in the existing table "cus_tbl".

ALTER TABLE cus_tbl
ADD cus_age **varchar**(40) NOT NULL;

```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_age varchar(40) NOT NULL;
Query OK, 3 rows affected (0.48 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT*FROM cus_tbl;
+--------+--------------+-------------+----------+
| cus_id | cus_firstname | cus_surname | cus_age |
+--------+--------------+-------------+----------+
|      5 | Ajeet         | Maurya      |          |
|      6 | Deepika       | Chopra      |          |
|      7 | Vimal         | Jaiswal     |          |
+--------+--------------+-------------+----------+
3 rows in set (0.00 sec)

mysql> _
```

Add multiple columns in the table

Syntax:

ALTER TABLE table_name
ADD new_column_name column_definition
[**FIRST** | **AFTER** column_name],
ADD new_column_name column_definition
[**FIRST** | **AFTER** column_name],

...
;

Example:

In this example, we add two new columns "cus_address", and cus_salary in the existing table "cus_tbl". cus_address is added after cus_surname column and cus_salary is added after cus_age column.

```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_address varchar(100) NOT NULL
    -> AFTER cus_surname,
    -> ADD cus_salary int(100) NOT NULL
    -> AFTER cus_age ;
Query OK, 3 rows affected (0.34 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

MODIFY column in the table

The MODIFY command is used to change the column definition of the table.

Syntax:

ALTER TABLE table_name
MODIFY column_name column_definition
[**FIRST** | **AFTER** column_name];

Example:

In this example, we modify the column cus_surname to be a data type of varchar(50) and force the column to allow NULL values.

```
mysql> ALTER TABLE cus_tbl
    -> MODIFY cus_surname varchar(50) NULL;
Query OK, 3 rows affected (0.33 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> DESCRIBE cus_tbl;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| cus_id       | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname| varchar(100) | NO   |     | NULL    |                |
| cus_surname  | varchar(50)  | YES  |     | NULL    |                |
| cus_address  | varchar(100) | NO   |     | NULL    |                |
| cus_age      | varchar(40)  | NO   |     | NULL    |                |
| cus_salary   | int(100)     | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

DROP column in table

Syntax:

ALTER TABLE table_name
DROP COLUMN column_name;

Let's take an example to drop the column name "cus_address" from the table

"cus_tbl".

```
mysql> ALTER TABLE cus_tbl
    -> DROP COLUMN cus_address;
Query OK, 3 rows affected (0.37 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_age | cus_salary |
+--------+---------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |         |          0 |
|      6 | Deepika       | Chopra      |         |          0 |
|      7 | Vimal         | Jaiswal     |         |          0 |
+--------+---------------+-------------+---------+------------+
3 rows in set (0.00 sec)
```

RENAME column in table

Syntax:

ALTER TABLE table_name
CHANGE **COLUMN** old_name new_name
column_definition
[**FIRST** | **AFTER** column_name]

Example:

In this example, we will change the column name "cus_surname" to "cus_title".

```
mysql> ALTER TABLE  cus_tbl
    -> CHANGE COLUMN cus_surname cus_title
    -> varchar(20) NOT NULL;
Query OK, 3 rows affected (0.25 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

MySQL TRUNCATE Table

- The TRUNCATE statement in MySQL removes the complete data without removing its structure.
- we use this command when we want to delete an entire data from a table without removing the table structure.

Syntax:

TRUNCATE TABLE table_name;

MySQL DROP Table

- MYSQL uses a Drop Table statement to delete the existing table.
- This statement removes the complete data of a table along with the whole structure or definition permanently from the database.

Syntax:

DROP TABLE table_name;

4.	**Write a short note the concept of sub queries with example.**	3
5	**Consider the database shown in Figure 1.2, whose schema is shown in Figure 2.1. What are the referential integrity constraints that should hold on the schema? Write appropriate SQL DDL statements to define the database.** **<u>DDL commands to define the database;</u>** **To create student table:** CREATE TABLE STUDENT (Name VARCHAR(30) NOT NULL, StudentNumber INTEGER NOT NULL, Class CHAR NOT NULL, Major CHAR(4), PRIMARY KEY (StudentNumber)); **To create course table:** CREATE TABLE COURSE (CourseName VARCHAR(30) NOT NULL, CourseNumber CHAR(8) NOT NULL, CreditHours INTEGER, Department CHAR(4), PRIMARY KEY (CourseNumber), UNIQUE (CourseName)); **To create table prerequisite table:** CREATE TABLE PREREQUISITE (CourseNumber CHAR(8) NOT NULL, PrerequisiteNumber CHAR(8) NOT NULL, PRIMARY KEY (CourseNumber, PrerequisiteNumber), FOREIGN KEY (CourseNumber) REFERENCES COURSE (CourseNumber),	3

FOREIGN KEY (PrerequisiteNumber) REFERENCES

COURSE (CourseNumber));

The referential integrity constraints to be placed on PREREQUISITE table are:

- The CourseNumber in the PREREQUISITE table should depend on the CourseNumber attribute in the COURSE table. Hence the CourseNumber should be the foreign key in the PREREQUISITE table.
- The PrerequisiteNumber in the PREREQUISITE table should also depend on the CourseNumber attribute in the COURSE table. Hence the PrerequisiteNumber should be the foreign key in the PREREQUISITE table.

To create SECTION table:

CREATE TABLE SECTION (SectionIdentifier INTEGER NOT NULL,

CourseNumber CHAR(8) NOT NULL,

Semester VARCHAR(6) NOT NULL,

Year CHAR(4) NOT NULL,

Instructor VARCHAR(15),

PRIMARY KEY (SectionIdentifier),

FOREIGN KEY (CourseNumber) REFERENCES

COURSE (CourseNumber));

The referential integrity constraints to be placed on SECTION table are:

- The CourseNumber in the SECTION table should depend on the CourseNumber attribute in the COURSE table. Hence the CourseNumber should be the foreign key in the SECTION table.

To create GRADE_REPORT table:

CREATE TABLE GRADE_REPORT (StudentNumber INTEGER NOT NULL,

SectionIdentifier INTEGER NOT NULL,

Grade CHAR,

PRIMARY KEY (StudentNumber, SectionIdentifier),

FOREIGN KEY (StudentNumber) REFERENCES

STUDENT (StudentNumber),

FOREIGN KEY (SectionIdentifier) REFERENCES

SECTION (SectionIdentifier));

	The referential integrity constraints to be placed on GRADE_REPORT table are: ● The StudentNumber in the GRADE_REPORT table should depend on the StudentNumber attribute in the STUDENT table. Hence the StudentNumber should be the foreign key in the GRADE_REPORT table. ● The SectionIdentifier in the GRADE_REPORT table should depend on the SectionIdentifier attribute in the SECTION table. Hence the SectionIdentifier should be the foreign key in the GRADE_REPORT table.	
6	**Explain various relational operations. Show the results of the following operations:** a. $T1 \bowtie_{T1.P = T2.A} T2$ b. $T1 \bowtie_{T1.Q = T2.B} T2$ c. $T1 \bowtie_{T1.P = T2.A} T2$ d. $T1 \bowtie_{T1.Q = T2.B} T2$ e. $T1 \cup T2$ f. $T1 \bowtie_{(T1.P = T2.A \text{ AND } T1.R = T2.C)} T2$ **TABLE T1** P Q R 10 a 5 15 b 8 25 a 6 **TABLE T2** A B C 10 b 6 25 c 3 10 b 5 **(a)** **P Q R A B C** **10 a 5 10 b 6** **10 a 5 10 b 5** **25 a 6 25 c 3** **(b)** **P Q R A B C** **15 b 8 10 b 6** **15 b 8 10 b 5**	3

	(c) **P Q R A B C** **10 a 5 10 b 6** **10 a 5 10 b 5** **15 b 8 null null null** **25 a 6 25 c 3** **(d)** **P Q R A B C** **15 b 8 10 b 6** **null null null 25 c 3** **15 b 8 10 b 5** **(e)** **P Q R** **10a 5** **15 b 8** **25 a 6** **10b 6** **25 c 3** **10b 5** **(f)** **P Q R A B C** **10 a 5 10 b 5**	
7	**Define is join? Differentiate various join operations in SQL with examples? .**	4
8	**Consider the following view, DEPT_SUMMARY, defined on the COMPANY Database**	5

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 3.5
Schema diagram for the COMPANY relational database schema.

CREATE VIEW DEPT_SUMMARY (D, C, Total_s, Average_s)
AS SELECT Dno, COUNT (*), SUM (Salary), AVG (Salary)
FROM EMPLOYEE
GROUP BY Dno;
State which of the following queries and updates would be allowed on the view. If a query or update would be allowed, show what the corresponding query or update on the base relations would look like, and give its result when applied to the database in Figure 3.6.

a. **SELECT * FROM DEPT_SUMMARY;**
b. **SELECT D, C FROM DEPT_SUMMARY WHERE TOTAL_S > 100000;**
c. **SELECT D, AVERAGE_S FROM DEPT_SUMMARY WHERE C > (SELECT C FROM DEPT_SUMMARY WHERE D=4);**
d. **UPDATE DEPT_SUMMARY SET D=3 WHERE D=4;**
e. **DELETE FROM DEPT_SUMMARY WHERE C > 4;**

9 | **Explain different Aggregate functions in SQL with examples?** | 2

10 | **Write SQL update statements to do the following on the database schema shown in Figure 1.2.**
a. Insert a new student, <‘Johnson’, 25, 1, ‘Math’>, in the database.
b. Change the class of student ‘Smith’ to 2.
c. Insert a new course, <‘Knowledge Engineering’, ‘CS4390’, 3, ‘CS’>.
d. Delete the record for the student whose name is ‘Smith’ and whose student number is 17.

a. **Insert a new student, <‘Johnson’, 25, 1, ‘Math’>, in the database.**
- MySQL INSERT statement is used to store or add data in MySQL table within the database.

Syntax:

The below is generic syntax of **SQL INSERT INTO** command to insert a single | 6

record in MySQL table:

INSERT INTO table_name (field1, field2,...fieldN)

VALUES

(value1, value2,...valueN);

- To insert new student in the database the query is
- INSERT INTO STUDENT VALUES (“Johnson”, 25, 1, “MATH”)

b. Change the class of student ‘Smith’ to 2.
MySQL UPDATE query is a DML statement used to modify the data of the MySQL table within the database.

Syntax

Following is a generic syntax of UPDATE command to modify data into the MySQL table:

UPDATE table_name
SET column_name1 = new-value1,
column_name2=new-value2, ...
[**WHERE** Clause]

- table_name:It is the name of a table in which we want to perform updation.
- **column_name:**It is the name of a column in which we want to perform updation with the new value using the SET clause. If there is a need to update multiple columns, separate the columns with a comma operator by specifying the value in each column.
- WHERE Clause: It is optional. It is used to specify the row name in which we are going to perform updation. If we omit this clause, MySQL updates all rows.

To update the class of a student the query is

UPDATE STUDENT SET CLASS = 2 WHERE Name=”Smith”

c. Insert a new course, <‘Knowledge Engineering’, ‘CS4390’, 3, ‘CS’>.

- MySQL INSERT statement is used to store or add data in MySQL table within the database.

Syntax:

The below is generic syntax of **SQL INSERT INTO** command to insert a single record in MySQL table:

INSERT INTO table_name (field1, field2,...fieldN)
VALUES
(value1, value2,...valueN);

To insert a new course the query is
INSERT INTO COURSE VALUES (“Knowledge Engineering”,”CS4390”, 3,”CS”)

	d. Delete the record for the student whose name is 'Smith' and whose student number is 17. MySQL DELETE statement is used to remove records from the MySQL table that is no longer required in the database. This query in MySQL deletes a full row from the table DELETE FROM table_name WHERE condition; ● In the above statement, we have to first specify the table name from which we want to delete data. ● Second, we have to specify the condition to delete records in the WHERE clause, which is optional. ● If we omit the WHERE clause into the statement, this query will remove whole records from the database table. To delete the record from the student the query is DELETE FROM STUDENT WHERE Name='Smith' AND StudentNumber=17	

QNo	Question	BTL
1.	**Define normalization and explain 1st and 2nd normal forms with an example in detail.**	**2**
2.	**A) Explain informal Design Guidelines for relational schema.** **B) Explain the concept of functional dependency in normalization.**	**1**
3.	**List and explain the properties of relational decomposition.**	**2**
4.	**Describe the concept of query processing and optimization with a neat sketch.**	**2**
5.	**Consider the following relation:** **CAR_SALE(Car#, Date_sold, Salesperson#, Commission%, Discount_amt)** **Assume that a car may be sold by multiple salespeople, and hence {Car#, Salesperson#} is the primary key. Additional dependencies are** Date_sold → Discount_amt and Salesperson# → Commission%	**3**

	Based on the given primary key, is this relation in 1NF, 2NF, or 3NF? Why or why not? How would you successively normalize it completely?	
6.	**All relations in BCNF are said to be in 3NF but reverse does not hold true. Justify the statement with a suitable example.**	**5**
7.	**Explain multivalued dependency in 4NF with an example.**	**2**
8.	**Explain join dependency in 5NF with an example.**	**2**
9.	**Consider the following relations for an order-processing application database at ABC, Inc.** **ORDER (O#, Odate, Cust#, Total_amount)** **ORDER_ITEM(O#, I#, Qty_ordered, Total_price, Discount%)** **Assume that each item has a different discount. The Total_price refers to one item, Odate is the date on which the order was placed, and the Total_amount is the amount of the order. If we apply a natural join on the relations ORDER_ITEM and ORDER in this database, what does the resulting relation schema look like? What will be its key? Show the FDs in this resulting relation. Is it in 2NF? Is it in 3NF? Why or why not?**	**3**
10.	**Differentiate between 4NF and 5NF.**	**4**

QNo	Question	BTL
1.	**Describe the ACID properties of a transaction with example.**	**2**
2.	**Define and explain Transaction serializability and its types with example?**	**1**
3.	**Discuss various Transaction Isolation Levels and its implementation.**	**2**
4.	**Explain the concept of two phase locking techniques for concurrency control.**	**2**
5.	**Describe about the concurrency control by timestamp ordering.**	**2**
6.	**a) Write a short note on Data recovery techniques.** **b) Explain the need of shadow paging in data recovery.**	**2**

7.		**5**

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

Change transaction T2 in Figure 21.2(b) to read
read_item(X);
X := X + M;
if X > 90 then exit
else write_item(X);
Discuss the final result of the different schedules in

	T_1	T_2	
	read_item(X); $X := X - N$;		
		read_item(X); $X := X + M$;	
ime	write_item(X); read_item(Y);		
		write_item(X);	←
	$Y := Y + N$; write_item(Y);		

	T_1	T_2	

	read_item(X);
	$X := X - N$;
	write_item(X);		

ime		read_item(X);
		$X := X + M$;
		write_item(X);	

	read_item(Y);		←

where M = 2 and N = 2, with respect to the following questions: Does adding the above condition change the final outcome? Does the outcome obey the implied consistency rule (that the capacity of X is 90)?

8.	**Check whether the given schedule S is view serializable or not-** **T1** | **T2** R (A) | A = A + 10 | | R (A) | A = A + 10 W (A) | | W (A) R (B) | B = B + 20 | | R (B) | B = B x 1.1 W (B) | | W (B)	**5**
9.	**Discuss about conflict serializability.Check whether the schedule is conflict serializable or not.** Transaction T1 | Transaction T2 R1 (A) | W1 (A) | | R2 (A) R1 (B) |	**3**
10.	**Differentiate between exclusive lock and shared lock with an example.**	**4**

QNo	Question	BTL
1.	**Define normalization and explain 1st and 2nd normal forms with an example in detail.**	**2**
2.	**A) Explain informal Design Guidelines for relational schema.** **B) Explain the concept of functional dependency in normalization.**	**1**
3.	**List and explain the properties of relational decomposition.**	**2**
4.	**Describe the concept of query processing and optimization with a neat sketch.**	**2**
5.	**Consider the following relation:** **CAR_SALE(Car#, Date_sold, Salesperson#, Commission%, Discount_amt)** **Assume that a car may be sold by multiple salespeople, and hence {Car#, Salesperson#} is the primary key. Additional dependencies are** Date_sold → Discount_amt and Salesperson# → Commission% **Based on the given primary key, is this relation in 1NF, 2NF, or 3NF? Why or why not? How would you successively normalize it completely?**	**3**

6.	**All relations in BCNF are said to be in 3NF but reverse does not hold true. Justify the statement with a suitable example**.	**5**
7.	**Explain multivalued dependency in 4NF with an example.**	**2**
8.	**Explain join dependency in 5NF with an example.**	**2**
9.	**Consider the following relations for an order-processing application data-base at ABC, Inc.** **ORDER (O#, Odate, Cust#, Total_amount)** **ORDER_ITEM(O#, I#, Qty_ordered, Total_price, Discount%)** **Assume that each item has a different discount. The Total_price refers to one item, Odate is the date on which the order was placed, and the Total_amount is the amount of the order. If we apply a natural join on the relations ORDER_ITEM and ORDER in this database, what does the resulting relation schema look like? What will be its key? Show the FDs in this resulting relation. Is it in 2NF? Is it in 3NF? Why or why not?**	**3**
10.	**Differentiate between 4NF and 5NF.**	**4**