Final (2019-20):

1(a): What is ~~database~~ (DBMS)? List and explain four reasons why DBMS is used instead of file processing system.

• DBMS: DBMS is a software system that allows users to define, create, maintain, and control access to a database. It manages data efficiently and allows safe, consistent access by multiple users.

• Four reason:

– Data ~~de~~ redundancy control: In file system the same data might be stored in multiple files. DBMS eliminates duplication by centralizing data, ensuring data is stored only oneee.

– Data integrity and consistency: DBMS enforces rules to ensure data ~~to~~ is accurate and consistent. File systems can not ensure that all files have up-to-date or valid data.

– Data security: DBMS provides user-based access control – different users can have different permissions. File system don't offer this level of control.

– Easy data access and querying: DBMS uses query languages like SQL to retrieve and manipulate data efficiently. In file systems, data retrieval is manual and slower.

1(b): R = (M, N, P, S, T)

1) Super Key: multiple (MN, MP, MS, MT, NP, NS, NT, PS, PT, ST, MNP, ~~MPS~~ MNS, MNT, STP, SPN, MNPS, MNPT, MNPST)

2) Candidatae Key: 2 (MN, NS)

3) primary Key: 1 (can be either 'MN' or NS depends on user).

1(c) How many attributes you can use in a table? Is there any limitations? Why you need to split the attributes in multiple tables?

The number of attributes in a table depends on the DBMS.

Example limits:

- MySQL: up to 4096 columns per table
- PostgreSQL: up to 1600 columns per table.
- SQL server: up to ~~21~~ 1024 Columns
- Oracle: up to 1000 columns per table.

So, yes there is a limitation, but it depends on the database system and row size.

- Split attributes into multiple tables:

1. Remove Redundency: prevents duplicate data.
2. Ensure data consistency: when data is update in one place, changes reflect everywhere.
3. Improve data Integrity: Foreign key constraints ensure valid and related data.
4. Better query performance: smaller tables mean faster queries and less memory usage.
5. Scalability and maintainability: Easier to update and maintain smaller tables than one huge with hundreds of columns.

1(d) Keyword queryes in Web search are quite different from database queries. List ke difference between the two, in terms of way the queries are specified and in terms of what is the result of a query.

Way queries are specified:

Feature	Web search	Database Queries
Language	Natural language / Keywords	Structured query language
Syntax	Informal, no strict syntax	Formal syntax
User Knowledge	User often doesn't Know data structure.	User must Know Schema
Query Example	best scientific movie 2025	SELECT * FROM Movies WHERE Genre = 'Sci-Fi' AND Year = 2025;

Result of a Query:

Feature	Web search	Database Query
Type of result	Ranked list of documents/ Web pages.	Exact rows from a structured database
Nature of output	Unstructured or semi-structured (text, link)	Structured data
Ranking	Results ranked by relevance	No ranking, exact match
Uncertainty	Results may be approximate or fuzzy	Results are deterministic and precise.

Web search is designed for human understanding, uses keywords, and gives relevant documents. Database queries are for data retrieval, use structured syntax, and return exact data.

2(a) What is database trigger? Discuss the strength and weakness of the trigger mechanism.

A trigger is a special kind of stored procedure in a database that automatically executes in response to certain events on a table or view.

Strength of trigger mechanism:

- Automation: Automatically enforces rules or actions without manual input.
- Consistency: Ensures data integrity across related tables.
- Centralized logic: Business rules are stored in one place, not repeated in multiple applications.
- Immediate response: Executes instantly after the specified event.
- Security: can enforce complex security constraints beyond basic privileges.

Weakness of trigger mechanism:

- Hidden logic: Triggers run in the background; developers may forget they exist, causing confusion.
- Debugging difficulty: Hard to trace and debug, especially when multiple triggers are chained.
- Performance overhead: Extra processing can slow down large INSERT, UPDATE or DELETE operations.
- Complexity: Overuse can make the system hard to maintain and understand.
- Limited portability: Trigger syntax and behaviour can differ across DBMS.

2(b) We can convert any weak entity set to a strong entity set by simply adding appropriate attributes. Then, why do we have weak entity sets.

We can convert any weak entity set to a strong entity set to a strong entity set by simply adding appropriate attributes. (Like a primary key). Weak entity sets are still used in database design for the following important reasons:

- Real world dependency: Weak entities represent objects that depend on another entity for their identification. For example, a Dependent of an Employee cannot be Uniquely identified without the Employee.
- Avoiding artificial Key: Creating Unique attributes may be unrealistic or unnecessary. Weak entities allows us to use a partial key combined with the primary key of the owner entity.
- Semantic clarity: Using weak entity helps to clearly express the existence dependency in the ER model. It shows that the entity can not exist independently.
- Maintaining referential integrity: Weak entities ensure that no such entity exists without its related strong entity. This simplifies database constraints and helps maintain data consistency.
- Better ER design and normalization: Weak entity helps to design a more normalized and structured database that avoids data redundancy.

2(c) Explain the distinct between database schema and instance.

Database schema	Database instance
- The overall structure or design of a database -	- The actual data stored in the database at a moment
- It is permanent	- It is Temporary
- It's like a blueprint of a building.	- It's like furniture and people currently in the building
- Defines tables, attributes types, relationships	- The current rows in the table.

2(d)

A file processing system has several disadvantages When used for a large, complex platform like a video site. Here's how these affect both actual video data and metadata.

1. Data Redundancy and Inconsistency:
   - Actual video data: If the same video is copied in different folders for different purposes, storage is Wasted.
   - Metadata: User information or tags might be stored in multiples. If updated in one place but not others, it causes inconsistency.
2. Lack of data sharing and Isolation:
   - video data: Multiple user may not be able to access or stream the same video efficiently.
   - Metadata: Different applications may find it difficult to access metadata uniformly.
3. Difficulties in Data Access:
   - video Data: Accessing specific video parts becoms difficult without structure.

• Metadata: Searching videos by title, tags, or uploader becomes slow and complex using plain files.

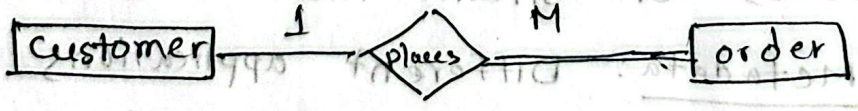
4. Integrity problems:
   • Video Data: File corruption might not be detected automatically.
   • Metadat: Ensuring values is difficult without constraints.

5. Security issues:
   • video data: Unauthorized users might access, modify or delete videos.
   • metadata: priavate user data can be leaked without proper access control.

3(a). ER-diagram:

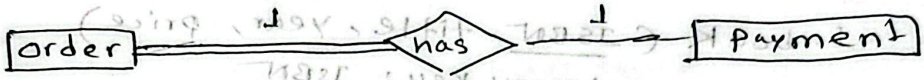
• Entities and Attributes:
   1. Book: (ISBN, title, year, price, category)
   2. customer (customer_id, name, email, Phone, address)
   3. Order (order_id, Date, Quantity, total_amount)
   4. payment (payment_id, payment_method, payment_status)
   5. Interest (preferred_category).

• Relationships and cardinality and participation:
   1. customer places order.



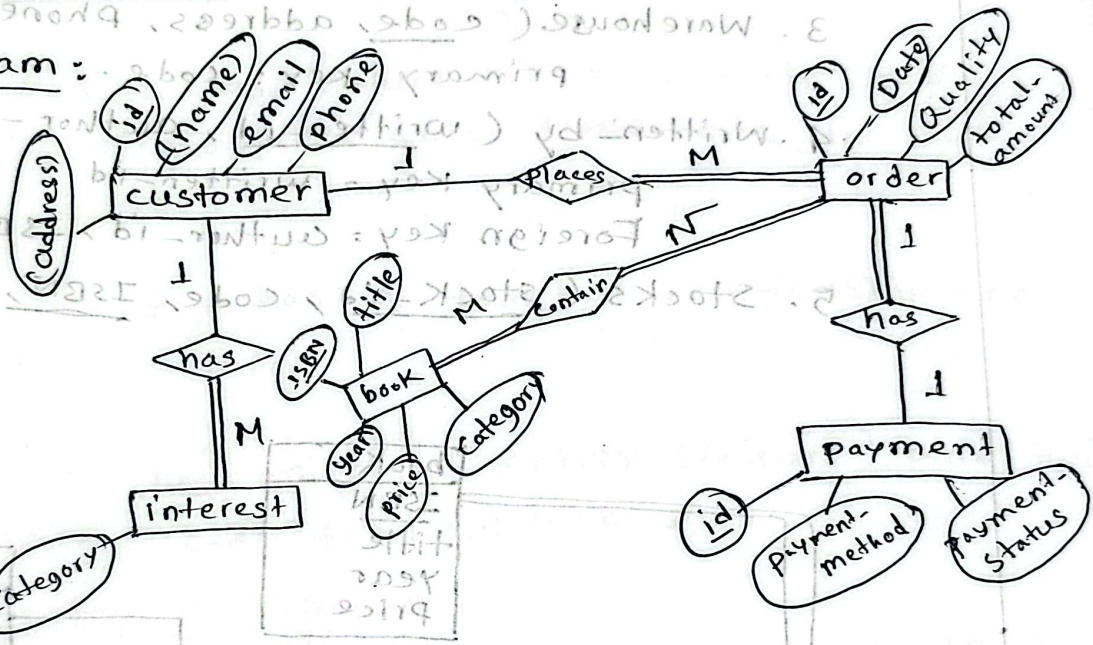
2. Order contains book

3. Order has payment:



4. Customer has interest:

ER- diagram:



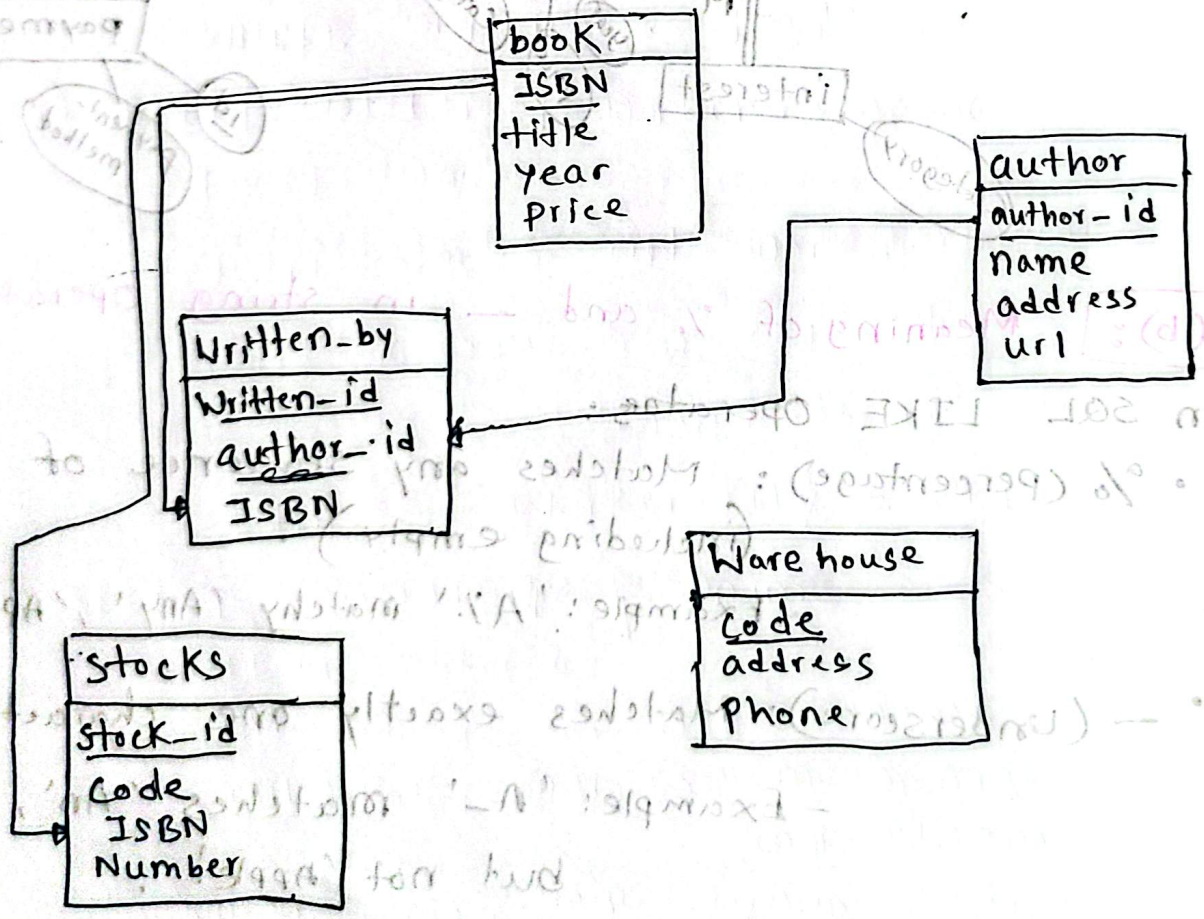
3(b): Meaning of % and _ in string operator:

In SQL LIKE Operator:

- % (Percentage): Matches any sequence of characters (including empty).
  - Example: 'A%' matchy 'Any', 'Apple', 'A' etc
- _ (Underscore): Matches exactly one character.
  - Example: 'A_' matches 'An', 'Ax' but not 'Apple'.

3(e) Schema Diagram:

1. book (ISBN, title, year, price)
   primary key: ISBN
2. author (author_id, name, address, url)
   primary key: author_id
3. Warehouse (code, address, phone)
   primary key: code
4. Written_by (written_id, author_id, ISBN)
   primary key = written_id
   Foreign key = author_id, ISBN
5. Stocks (stock_id, code, ISBN, number)



4(a) : Compares ER Model A and B.

(i) False

- Model A only shows a lecturer teaches a course → it does not show multiple lectures.
- Model B allows separate lecture entities, so multiple lectures are possible.

(ii) True

- Since Model B has lecture as a separate entity. We can store info like date, time etc for each lecture.

(iii) True.

- Model A only has a teaches relationship - no specific lecture info.

(iv) True.

- Model B has an extra entity (lecture) and two relationships → leads to more tables.

4(b) Entities and Relationships:

1. Lot (LotNumber, creatDate, ControlMaterial, LeftByUser).
2. productionUnits (Serial#, exactWeight, productType, QualityTest, productDate).
3. RawMaterials (MaterialID, Name, Type, Unitcost)
4. Includes (many to many between Lot and productionUnits)
5. createdForm (Many to many between productionUnits and Raw materials)

4(c) composite attributes:

In RawMaterials, the attribute 'Name' could be composite if it stores 'full material description like: 'MaterialName – Brand – Code".

• Explanation: If we break name into Material Name, Brand, code, it Becoms a composite attribute.

4(d) Identify Data Redundency:

In productionUnits, if productType, QualityTest, and productionDate are repeated for each unit — even if they belong to the same product batch — that's redundant.

• Reason: These attributes could be moved to a seperate product entity, reducing repetition.

5(a)

(i)
```
SELECT Wname
FROM Works
WHERE orgname = 'google';
```

(ii)
```
SELECT DISTINCT Worker.Wname
FROM Worker
JOIN Works ON Worker.Wname = Works.Wname
JOIN organization ON Works.orgname =
                     organization.orgname
WHERE Worker.city = 'Organization.city';
```

(iii)
```
SELECT DISTINCT e.Wname
FROM Worker e
JOIN manages m ON e.Wname = m.Wname
JOIN Worker mgr ON m."manager-name" =
                   mgr.Wname
WHERE e.city = mgr.city AND e.street = mgr.street;
```

(iv)
```
UPDATE Works
SET salary = salary * 1.075
WHERE Wname IN (SELECT DISTINCT
                "manager-name" FROM manages);
```

(v)
```
SELECT Wname
  FROM Worker
  WHERE city = (
        SELECT city
          FROM Worker
          WHERE Wname = 'Rahim');
```

(vi)
```
SELECT orgname
  FROM Works
  GROUP BY orgname
  ORDER BY COUNT (DISTINCT Wname) DESC
     LIMIT 1;
```

(vii)
```
CREATE VIEW HighEarners AS
  SELECT Wname, Salary
    FROM Works
    WHERE salary > (
          SELECT AVG (salary)
            FROM Works);
```

(viii)
```
SELECT Wname
  FROM Works
    WHERE orgname = 'FaceBook'
      AND jdate <= CURRENT-DATE -INTERVal
                   '5 years';
```

5(b)

(i). The table has:

FOREIGN KEY (manager-name) REFERENCES manager (employee-name) ON DELETE CASCADE;

that means:

- If a manager (a row where employee-name = some manager's name) is deleted,

then all the employees who had that person as their manager (rows where manager-name = that name) will also be deleted automatically.

- This is called cascading delete. It ensures no employee is left pointing to a non-existent manager.

(ii). If the statement was:

FOREIGN KEY (manager-name) REFERENCES manager (employee nam) ON DELETE RESTRICT;

then:

- If we try to delete a manager's record,
- The delation will fail if any employee still references that manager in their manager-name

= No deletion is allowed unless all dependent employees are first remored or reassigned.

(6) [a] given table:
- books (accessionno, isbn, title, author, publisher).
- users (userid, name, deptid, deptname);

given FDs:
accessionno → isbn
isbn → title
isbn → author
isbn → publisher.
userid → name
userid → deptid
userid → deptname

already 1NF, 2NF done.
For 3 NF:
Create following tables:

Books (accessionno, isbn)
ISBN_info (isbn, title, author, publisher)
users (userid, name, deptid).
Departments (deptid, deptname).

BCNF

[6'b] (i) AB → C
Row 1 and Row 3: A = 1, B = 2
Row 1 : C = 3, Row 3 : C = 4 → X Not equal → Doesn't hold.

(ii) B → D
Row 1 and 3: B = 2 → D = 4 (correct)
Row 2 : B = 4 → D = 4 (correct)
B → D holds.

(iii) DE → A
Row 1 and 2: D = 1, E = 5 → A = 1 (correct)
DE → A holds.

[6] [c] FDs: {AB → CD, BC → D}

Step 1: AB → C
AB → D
BC → D

Step 2: check AB → C
- can we remove A from AB?
  - B → C? No (not given)
- can we remove B?
  - A → C? NO
  - → so AB → C is minimal.

check AB → D
- Already BC → D exists. Try removing A.
- Is B → D or BC → D enough to get AB → D?

∴ minimal cover: {AB → C, BC → D}.

(7) [a] What is database transaction? Discuss Acid properties of database transaction.

⇒ A database transaction is a logical unit of work that contains one or more database operations which are executed as a single unit. It must be either fully completed or fully failed.

ACID properties:

- A (Atomicity): All operations in a transaction are completed; if one fails, the whole

transaction fails.

- C (consistency): The database remains in a consistent state before and after the transaction.
- I (Isolation): Transactions occur independently without interference.
- D (Durability): Once a transaction is committed, the changes are permanent even in the case of a crash.

7b

T_1 :	T_2 :
LOCK (A)	LOCK (A)
READ (A)	WRITE (A)
LOCK (B)	LOCK (B)
READ (B)	READ (B)
B = A+B	UNLOCK (A)
WRITE (B)	UNLOCK (B).
UNLOCK (A)	
UNLOCK (B)	

Is it deadlock free?

No, this may lead to a deadlock: If T_1 locks A and T_2 locks B at the same time, both will wait for each other to release the lock, leading to a circular wait. To prevent deadlock, transactions should lock resources in a consistent order:

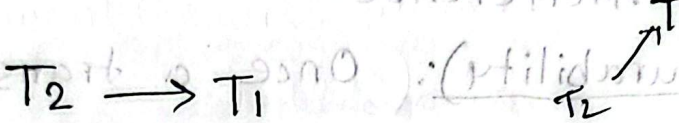
7c

S: R1(A); R2(A); R3(B); W1(A); R2(C); R2(B); W2(B); W1(C)

Check conflicting operations:

- R1(A) and W1(A) – same transaction, no conflict.

- R2(A) and W1(A) – conflict
- R3(B), R2(B), W2(B) – R2(B) and W2(B) is a conflict.
- W1(C) and R2(C) – conflict



No cycle in graph.

Q.N

- Dirty read: When one transaction reads data that has been written by another uncomitted transaction.

Example:

T1: UPDATE balance SET value = 500
T2: SELECT value (reads 500)
T1: ROLLBACK
→ T2 read a dirty value.

- Lost update: When two transactions read the same value and update it, one update is lost.

Example:

T1 = READ balance = 100 → balance = balance + 50 → write (150)

T2 = READ balance = 100 → balance = balance + 30 → write (130).

⇒ final value is 130, T1's update is lost.

8 a Differentiate among primary, secondary and clustering Indexing.

⇒ • primary index: Created on the primary key, entries are sorted according to key order.

• secondary index: created on non-primary attributes, may have duplicates.

• Clustering index: Based on a clustering field, physically arranges data in file order.

8 b When is it preferable to use a dense index rather than a sparse index?

⇒ Dense index is preferable when:

— Records are frequently accessed randomly.

— Records are small and uniformly distributed.

— For search heavy applications, as every record has an index.

8 c Is it possible in general to have two primary indices on the same relation for different Search keys?

⇒ No, ~~In one relat~~ in the same relation there cannot have two primary indices.

— A primary index is based on the primary key, which must be unique

- only one attribute can act as the primary key per relation.

8d

- course (couse_name, room, instructor)
- enrollment (course_name, student_name, grade)

Cluster file structure (Example):

course: DBMS, Room: 101, Instructor: Dr. Karim
→ students: [Tamim, A; Arif, B+; Nayeem, A-]

Couse: OS, Room: 202, Instructor: Dr. Rahman
→ students: [Mehedi, A+; Arif, A; Tamim, B]

Explanation: Here the file is clustered on course_name. The related enrollment records are physically stored with the Course information to improve I/O performance on querres like:

```
SELECT * FROM course JOIN
    enrollment USING (course_name).
```