note: 3.7 is not in syllabus.

Date: 3/10/24

DBMS → Database + Software

- it is a software to handling, storing, retrieving, manipulations of data.
- collection of related data
- contains tables called relations, rows called tuple, values called attributes.

Terminology.

SQL

* Syntax / Example to create table

```
Create table student (sname  varchar(20),
                      susn   varchar(25),
                      sage   not number(3));
```

use int

* Description of table.

```
DESC Student;
```

note: susn varchar(20) not null;
↳ the field should never be empty

* Insert

```
insert into student values ('xyz',
                            '1MS22CS0XX',
                            20)
```

note: varchar → alphanumeric.

* Read

```
select * from student;
```

(give all records of table)

Date : _____

select susn from student;
select * from student where susn = '20';

Add a field / column to table

* Alter Table → add (add a new field to table)
  → modify (change existing field name or datatype)

* Delete → remove record
* Drop → delete table
* update → set (change a record)

note: Initially,

create database msrit;
use database msrit;

note: show tables; (display tables existing in our database)

* Update syntax

update student set sname = 'acc' where susn = 'cs-64';

note: primary key → not null, unique values
foreign key → take value of primary key
it can be null.
Should have same datatype / domain as primary key
↓
datatype
size

Date : _____

* **Alter table add**

```
alter table student
add smarks int(3);
```

* **Commit** ⟶ used to permenantly commit to database.

* **Alter table rename**

```
alter table student rename to dept;
```

note: create table student (susn ~~in~~ varchar(10) primary key);

(OR)

```
( susn .varchar(10), primary key (susn));
```

* **Alter table drop**

```
alter table student drop column pincode ;
```

* **Alter table column name**

```
alter table student change sname stdname
                                varchar(30);
```

note: Data type is known as **domain**.

NOSQL has no restrictions in inserting records compared to SQL.

```
NOT IN (select rsid from res)
```

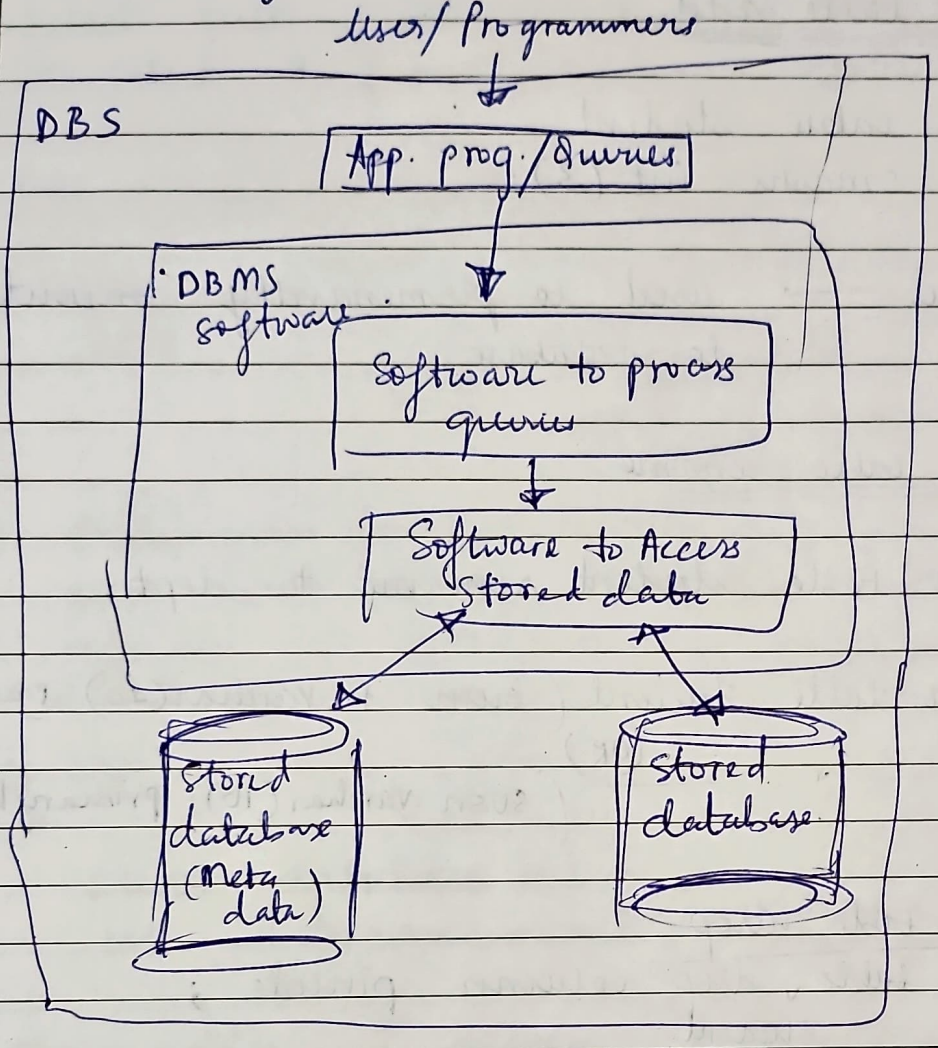
↓ minus

6 mark

14/10/24

Date : _____

Database system environment



* info abt data is meta data.

eg: Student table

the students values are in stored database.

the info abt the data stored in table like data type, freq., — meta data

* Software → Process the queries
  → Access the data stored.

Characteristic of database approach · processing

→ Database approach vs file approach.

- self describing nature
- support of multiple views of data
- data abstraction
- sharing data and multiuser.
- metadata is stored in system catalog.

Database users.

Actors on the scene.	Workers behind the scene.
* DBA (Database administrators)	* DBMS system designers and implementers
* Designer	* Tool developers
* end users. (4 parts)	* Operators and maintenance personnel.
↳ casual	
↳ parametic / naive	
↳ sophisticated	
↳ standalone.	
* system analysis & app. programmers (SW engineers)	

Advantages

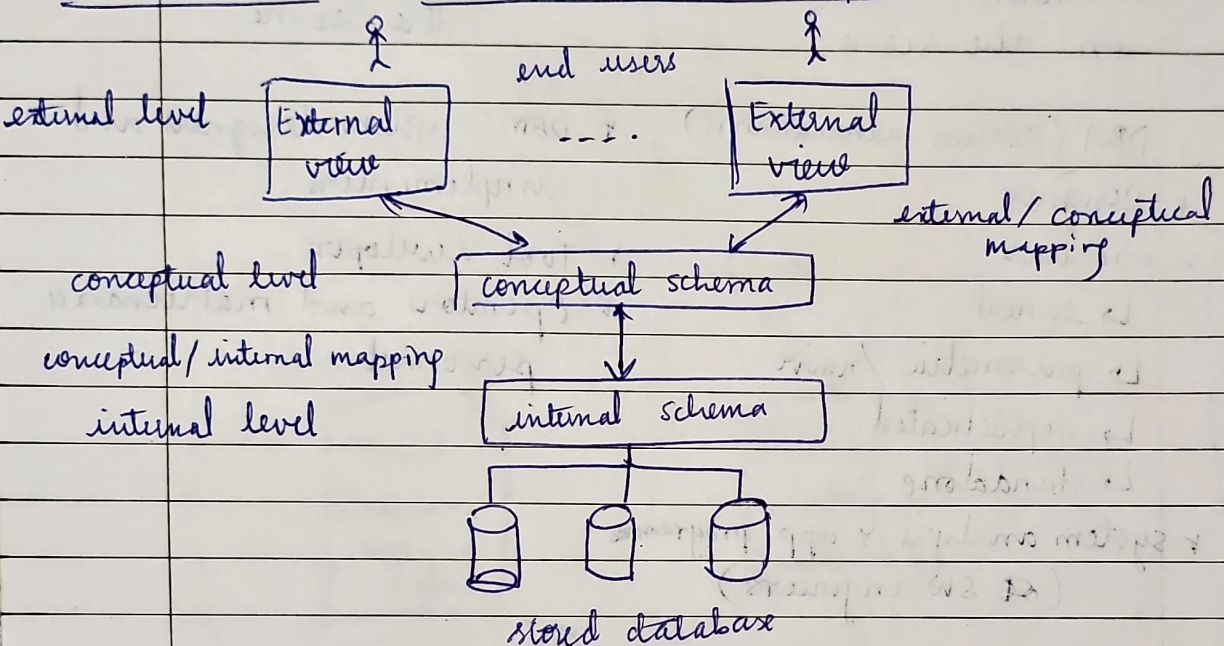
* Controlling redundancy.
* Restricted unauthorized access.
* Providing persistent storage for program objects
* B —"— storage structures and search techniques for efficient query processing
* Providing backup and recovery.
* —"— multiple user interface
* Representing complex relationships among data
* Enforcing integrity constraints
* Permitting inferencing and actions using rules.
* Addition implications of DB.
  - → potential for enforcing standards → availability of up to date info
  - → reduced app. dev. time → economics of scale.
  - → flexibility

Data Model (categories)

- high level or conceptual – entity, attributes and relationships – end users
- Representational or Implementation
- low level or physical – how data is stored in storage media

note: Schemas — description of database.

Three Schema architecture and data independence



Data independence

Logical

capacity to change conceptual schema w/o changing external

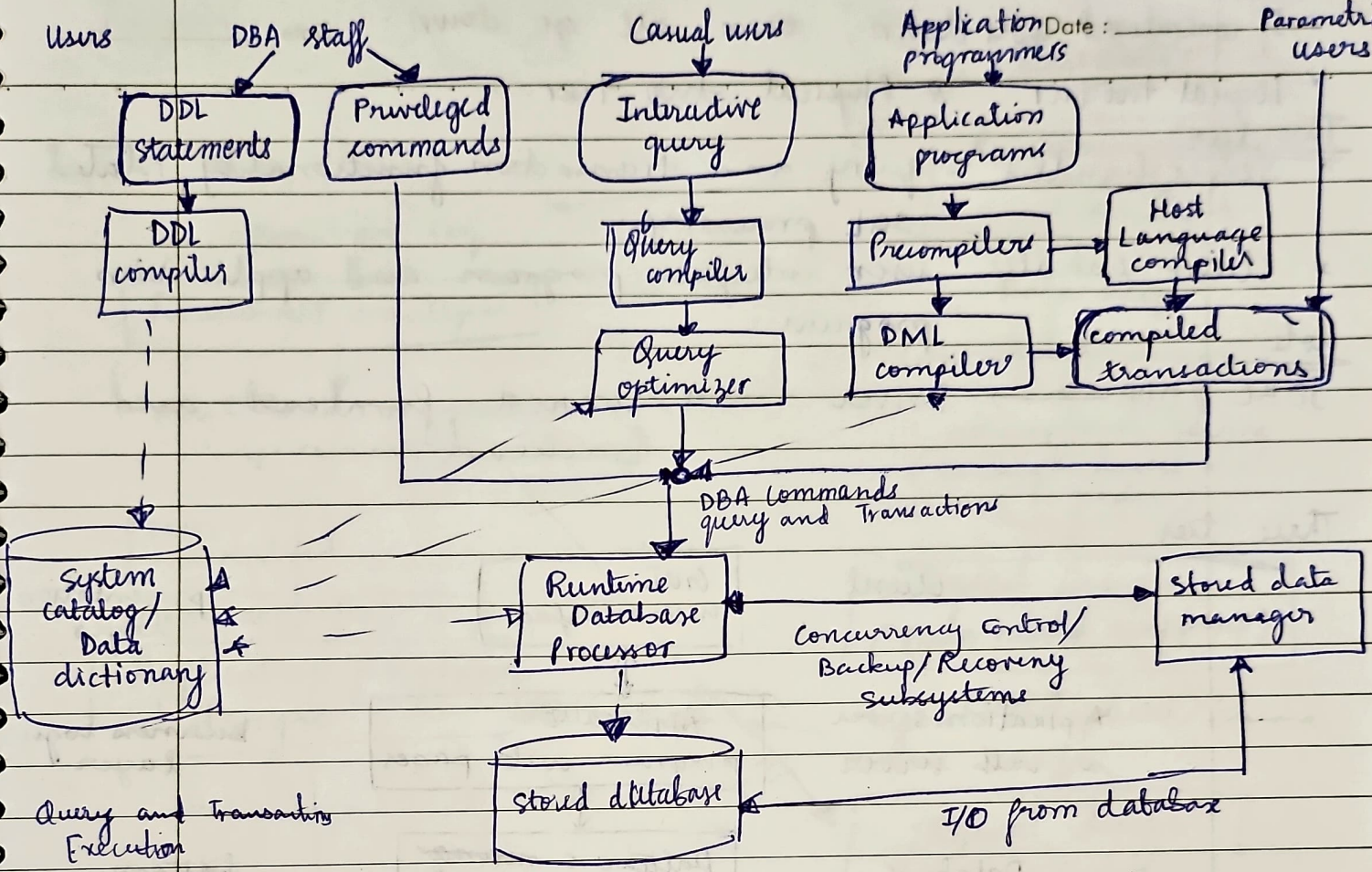
Physical

capacity to change internal schema w/o changing conceptual schema.

DBMS Interfaces

1. Menu based
2. Form based
3. Graphical User Interface
4. Natural Language Interface
5. Speech I/O
6. Parametric User Interfaces
7. Interfaces for DBA.

DBMS (component modules ~~(system environment)~~)



DDL compiler → stored meta data /process DDL (data def^{n} language)

Interactive query → interface for casual user.

query optimizer → rearrangement, eliminating redundancies, reordering of operations

~~When Not to Use a DBMS~~

note: Three components/parameter of computer system:
- Compute - CPU
- storage - RAM, HDD
- Networking capacity - Bandwidth.

When not to use DBMS

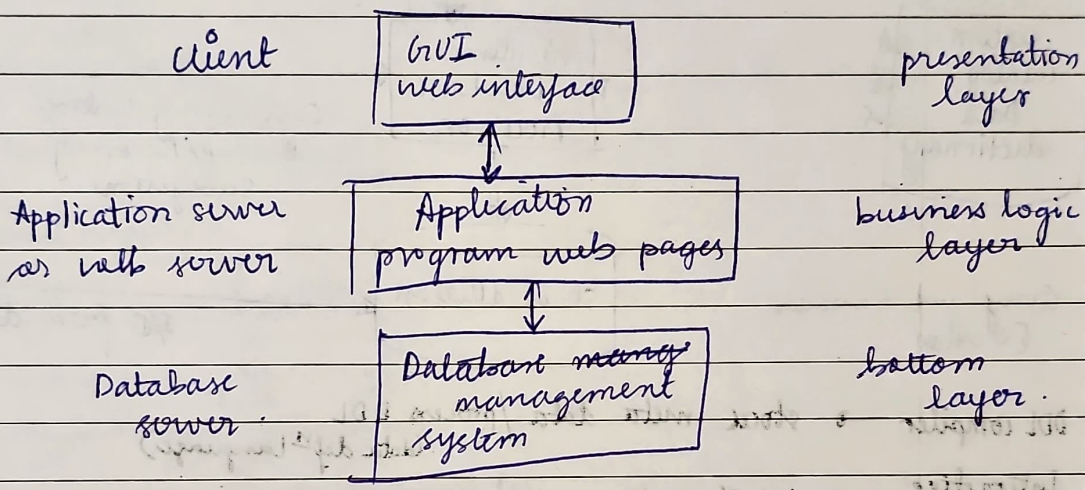
Centralized and client server architecture.

* If centralized goes down, then all go down.
* Logical two tier & Physical two tier.

Two tier
* Server handles - query and transaction functionality related SQL processing.
* client handles user interface program and application program.

note:
JDBC & ODB ⟶ Driver ⟶ Connect frontend and backend.

Three tier



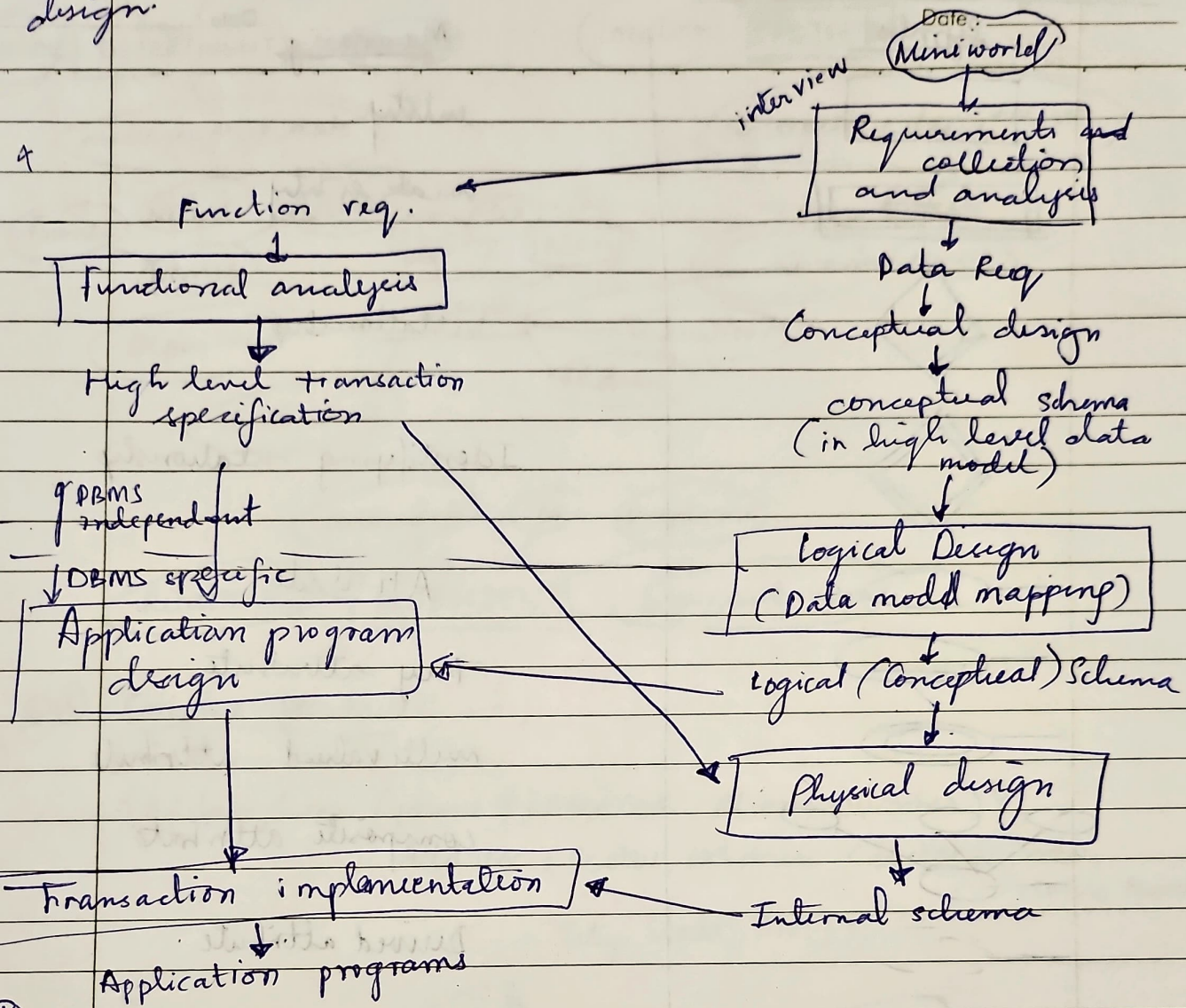
* Database capacity is separated as a separate server.

N-tier architecture
- multiple layers.

classification of DBMS

1. Data model - Relational, object, hierarchical & network, Native XML DBMS.
2. No. of users - single, multi
3. No. of sites - centralised, distributed
4. Cost - Open source, licensing
5. Types of access paths for storage files - or inverted file structure
6. General purpose / special purpose.

Using high level conceptual data model for database design.



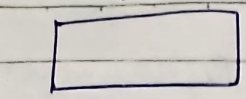
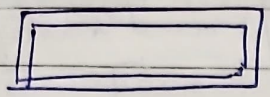
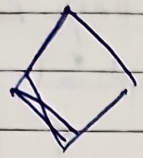
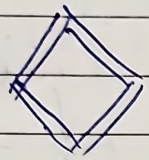
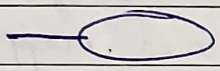
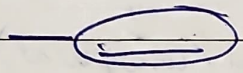
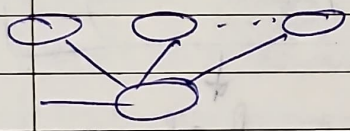
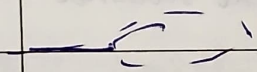
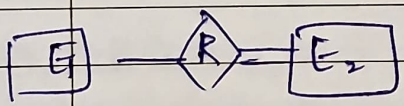
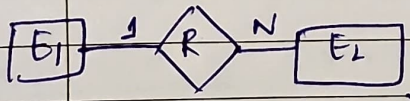
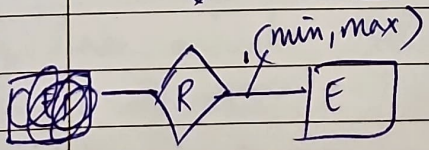
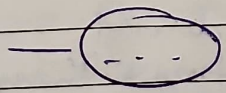
Types of attributes

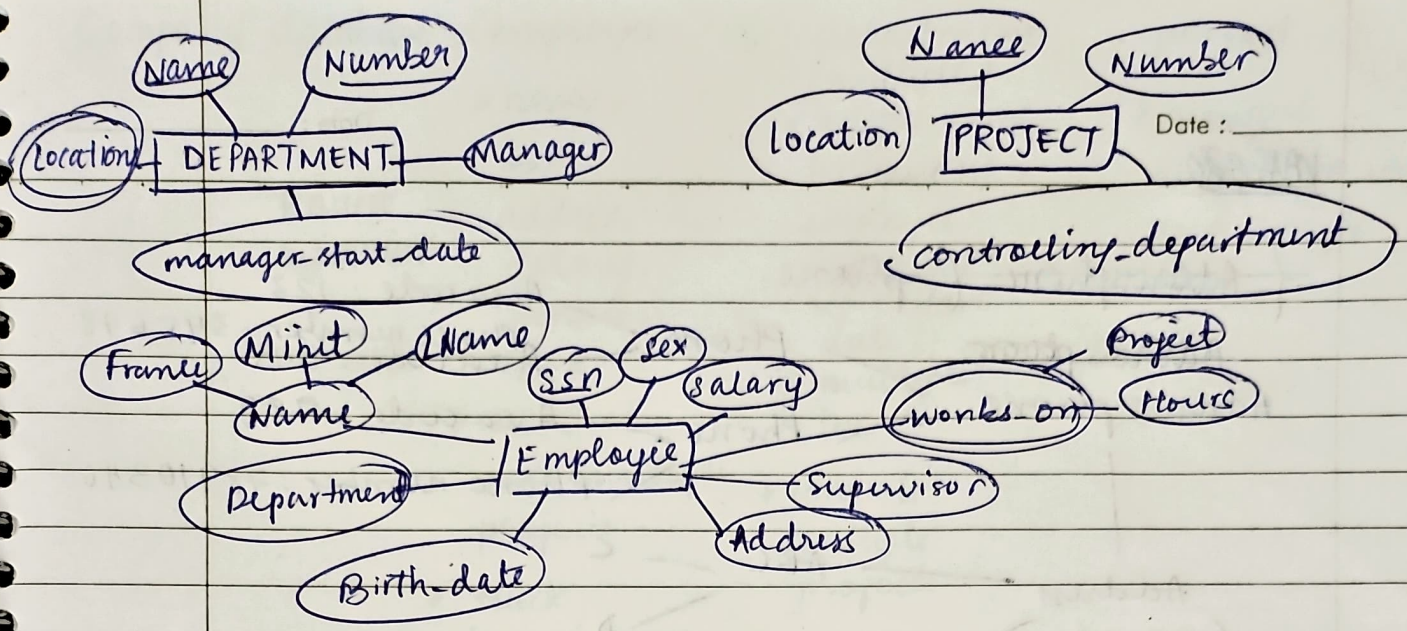
1. Composite vs Simple (Atomic)
2. Single valued vs multi valued
3. Stored vs derived
4. NULL values
5. Complex attributes

: [1] can be divided into subparts vs base attribute (can't divide)
[2] one value for a entity vs set of values for entity
[3] cannot be derived from other vs can be derived eg: Birthday age from birthday and today's date
[4] optional

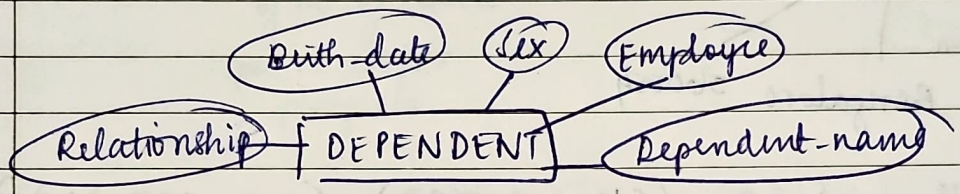
Notation for ER ~~dig~~ diagram:

Date : _____

Symbol :	Meaning
	entity
	weak entity
	relationship
	Identifying relationship
	Attribute
	Key attribute
	multivalued attribute
	composite attribute
	Derived attribute
	Total participation of E_2 in R.
	Cardinality Ratio 1 : N for $E_1 : E_2$ in R
	Structural Constraint (min, max) on Participation of E in R
	partial key.

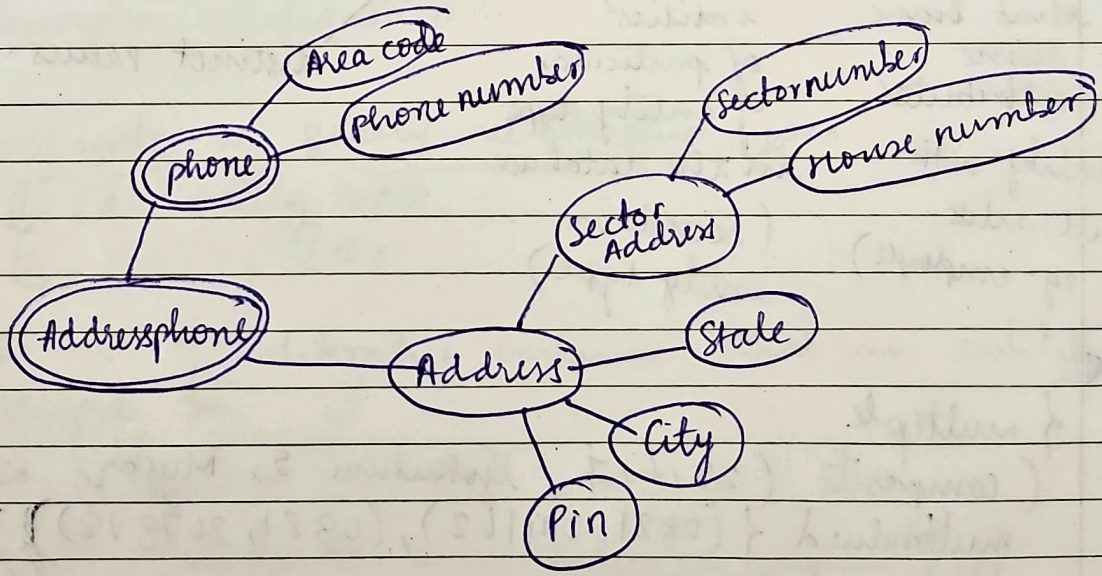


Date :



Complex Attribute

{Addressphone (phone {(Area Code, Phone number)},
Address (sector Address (sector Number, House Number), City, State, Pin))}



{Phonebook (~~phone~~ Address (Sectorno., house no., street name, city, state), phone {(Area code, no.)})}

Date : _____

~~ABC~~

~~{ Addressphone (phone~~

Address phone — Phone — Area code : 123
Phone number : 345678
~~Area code : 345~~

Address phone — Phone — Area code : 789
Phone number : 78910346.

Address — ABC — C
D

Address — Karnataka
Bengalore
560054.

{ Addressphone (phone ((123, 345678), (789, 783214)),
Address (ABC (3, 4), Bengalore, Karnataka,
560054)) }

→ Extension

Entity Types, Entity Sets, Keys and Values sets

↓ coll^n of entities that have same attributes (entity with attributes. eg: employee)

↓ coll^n of entities of particular of entity type in the database (records in entity type)

distinct values.

{ multiple
(composite (28, 6, 7, Gokulum 5, Mysore, Karnataka),
multivalued { (0821, 214162), (0821, 267892) }),
~~first address is one~~
(composite (32, 7, 3, Mathikere, Bangalore, Karnataka),
multivalued { (0821, 316273), (0821, 321421) })
}

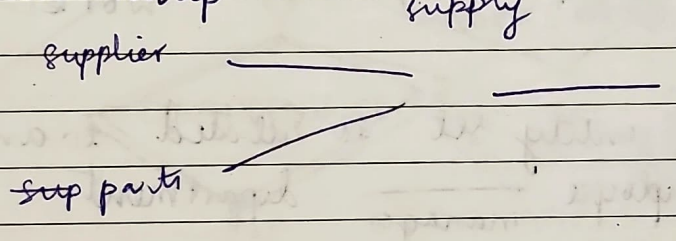
Company database – employee , department , project

employee
+ name, SSN, address, salary, gender, DOB.
+ direct supervisor
+ one dept but multiple projects
+ track of no. of hours per week for projects

department (multiple)
+ unique name, unique no., employee manager with start date, multiple loc.
+ controls multiple projects

project
+ unique name, ~~no~~ unique no and single loc.

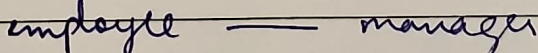
note: ~~Ter~~ Ternary relationship – 3 entity types in one.
relationship



note: Recursive relationship

Constraints on Binary relationship types
(1) Cardinality ratio
(2) participation

[1] Max. no. of relationship instances that an entity can participate in
↳ 1:1 (one to one).



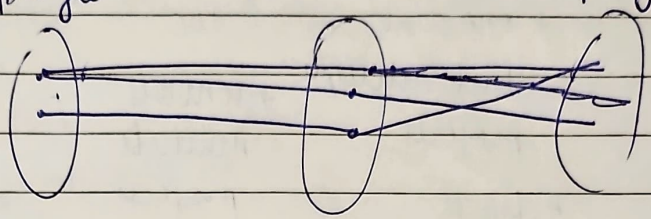
↳ 1:N (one to many)

↳ N:1 (many to one)

Date :

↳ M:N (many to many)

employee — work on — project



Participation - specifies whether existence of entity depends on its being related to another entity.

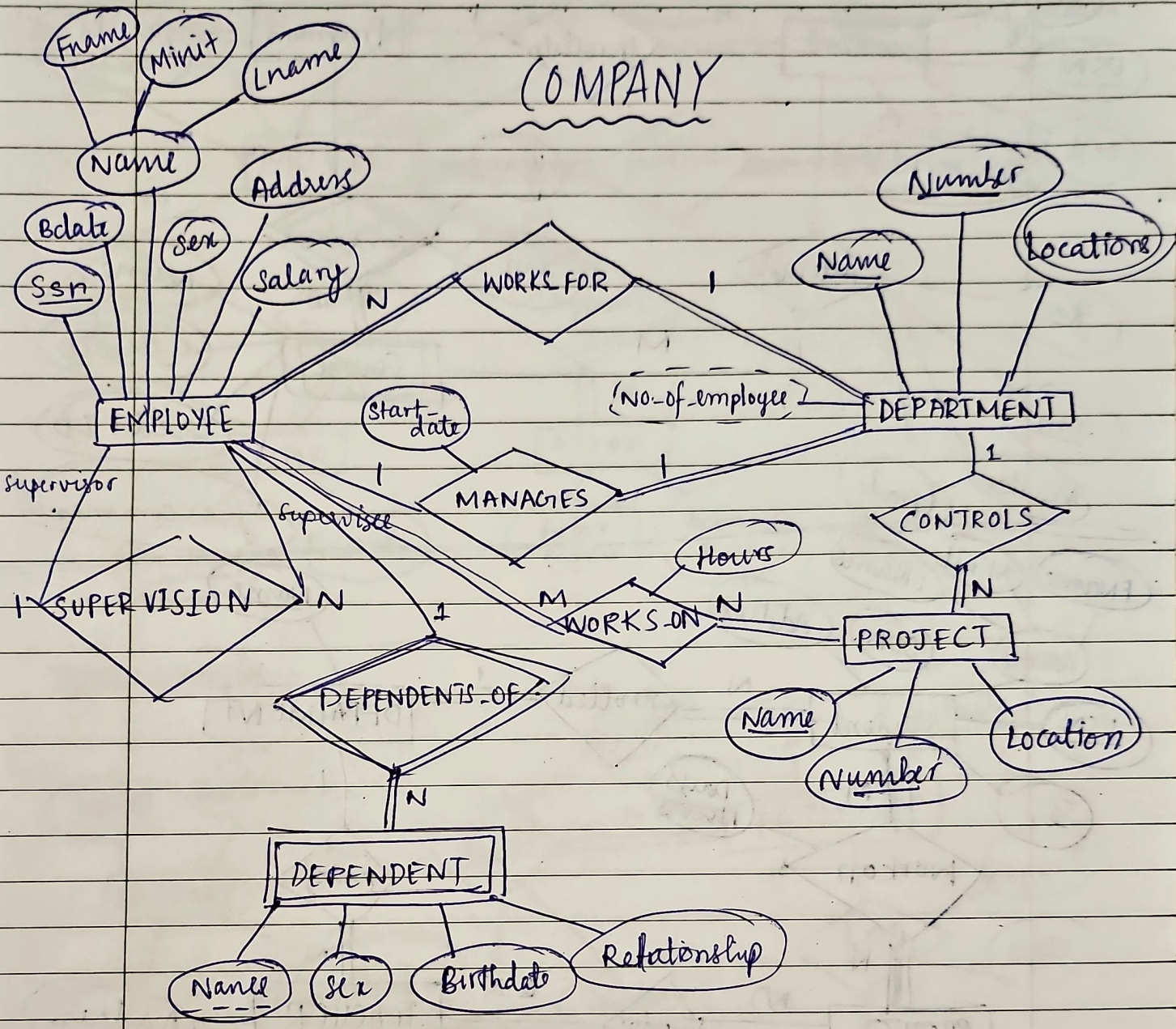
* Total -
every entity of an entity set must be related to another entity. eg: employee — department
works

* Partial
a part of entity set is related to another entity. eg. employee — department
manages

student
dept
teacher
projects

Date : _____

COMPANY



UNIT-2

Date : _____

(4) (*)

Characteristics of ~~database~~ relation.

(1) Ordering of tuples in relation
(2) Ordering of ~~tuples~~ values within tuple and an alternate definition of relation
(3) Values and NULLs in a tuple
(4) Interpretation of a relation

Relational Model Constraints

(1) Implicit constraints
(2) Explicit constraints
(3) Semantic constraints

Key constraints (no 2 tuples should be same)

* superkey $e_1(SK) \neq e_2(SK)$
* candidate key
* primary key
* key (minimal superkey)
* foreignkey

(*) COMPANY = (EMPLOYE, DEPT, DEPT_LOC, PROJECT, WORKS_ON, DEPEDENT).

EMPLOYEE

FName	MInit	LName	Ssn	Bdate	Address	Gender	Salary	Super_ssn	Dno

DEPT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOC

Dnumber	Dlocation

PROJECT

Pname	PNumber	Plocation	Dnum

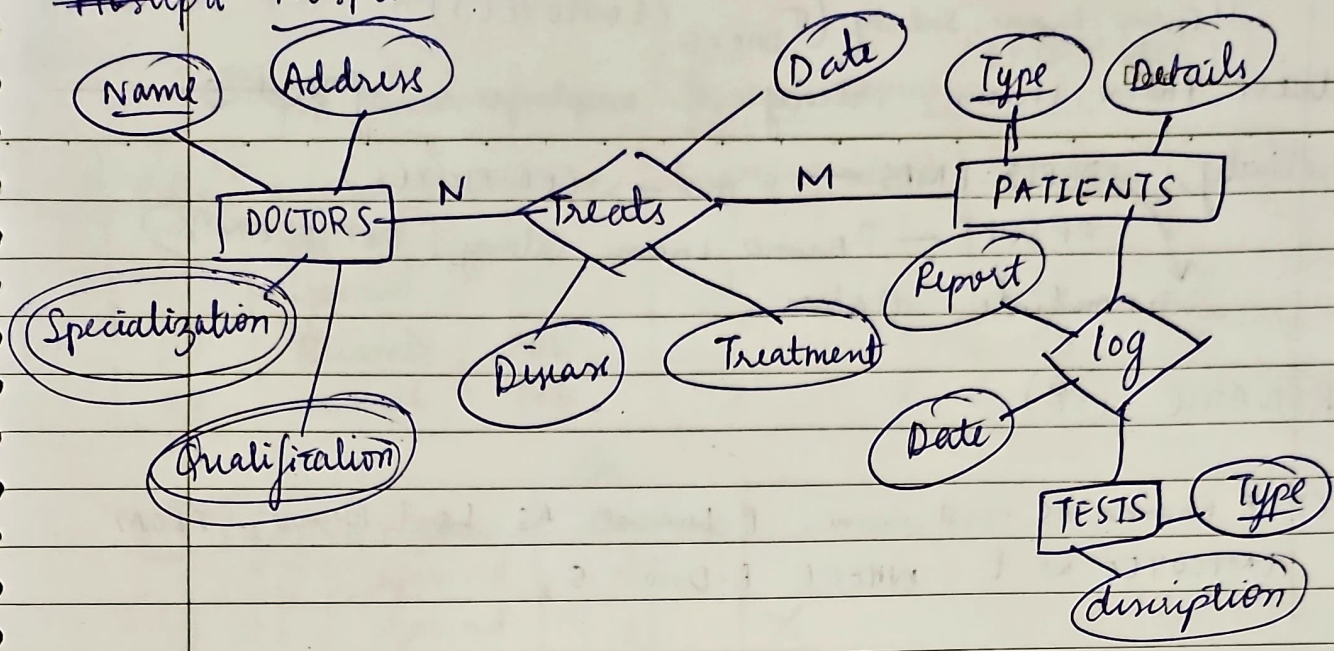
WORKS_ON

ESsn	Pno	Hours

DEPENDENT

ESsn	Dependent_name	Gender	Bdate	Relationship

~~Hosbpa~~ Hospital



Constraint Violations (8 mark)

Insert ① Domain constraint ② Key constraint ③ Entity integrity ④ Referential integrity

Delete ① Referential integrity

Update ① Referential integrity

Relational Algebra

[1] SELECT (σ)

Q) Select the employee tuples whose dept no. is 4.

$\sigma_{DNO=4}$ (EMPLOYEE)

Q) Select employee tuples whose salary > 30000

$\sigma_{SALARY>30000}$ (EMPLOYEE)

Q) select emp. work in dept 4 and >25000 or work in dept 5 and > 30000

$\sigma_{(DNO=4 \text{ AND } SALARY>25000)}$ OR $\sigma_{(DNO=5 \text{ AND } SALARY>30000)}$ (EMPLOYEE)

[2] PROJECT (Π)

Q) List each employee's first and last name

$\Pi_{Fname, Lname}$ (EMPLOYEE)

FName	Lname

x eliminates duplicates

SELECT DISTINCT Fname, Lname FROM EMPLOYEE

$\Pi_{Fname, Lname, Salary}(\sigma_{Dno=5}(EMPLOYEE))$

To retrieve Fname, Lname, salary of employee's of dept 5

Alternatively, DEPT5_EMPS $\leftarrow \sigma_{Dno=5}(EMPLOYEE)$

RESULT $\leftarrow \Pi_{Fname, Lname, salary}(DEPT5\_EMPS)$

intermediate relation

3] RENAME (ρ)

SELECT E.Fname AS First_name, E.Lname AS Last_Name FROM EMPLOYEE as E WHERE E.Dno = 5;

Cross Product / Cross join

Emp

EID	Ename	Salary	EDno
111	Govind	20K	1
112	Adarsh	15K	1
113	Bharath	20K	2

Dept

Dno	Dname
1	CS
2	IS

Emp X Dept

EID	Ename	ESalary	EDno	Dno	Dname
111	Govind	20K	1	1	CS
111	Govind	20K	1	2	IS
112	Adarsh	15K	1	1	CS
112	Adarsh	15K	1	2	IS
113	Bharath	20K	2	1	CS
113	Bharath	20K	2	2	IS

* Each row of emp table is combined with each row of dept table.

* Every tuple of first relation is joined with all tuples of second relation.

Q) Retrieve EID, ~~EMAIL~~ Ename, Dname whose salary > 16000.

$\Pi_{EID, Ename, Dname}(\sigma_{Esalary>16000}(Emp \times Dept))$

EID	Ename	Esalary	EDno	Dno	Dname	
111	Govind	20k	1	1	CS	
111	Govind	20K	1	2	IS	X
113	Bharath	20k	2	1	CS	X
113	Bharath	20k	2	2	IS	

$\Downarrow$

EID	Ename	Dname	
111	Govind	CS	
111	Govind	IS	X
113	Bharath	CS	X
113	Bharath	IS	

$\therefore \Pi_{EID, Ename, Dname}(\sigma_{Esalary>16000 \text{ AND } EDno = Dno}(Emp \times Dept))$

JOIN Operation.

Join operation:

$Emp \bowtie_{EDno = Dno} Dept$

~~EID Ename D.~~

EID	Ename	Esalary	EDno	Dno	Dname
111	Govind	20K	1	1	CS
112	Adarsh	15K	1	1	CS
113	Bharath	20K	2	2	IS

Q) T1

P	Q	R
10	a	5
15	b	8
25	a	6

A	B	C
10	b	6
25	c	3
10	b	5

Left outer join

① $T1 \bowtie T2$ $\quad T1.P = T2.A$

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	10	b	5
25	a	6	25	c	3
15	b	8	NULL	NULL	NULL

② $T1 \bowtie T2$ $\quad R = C$

P	Q	R	A	B	C
10	a	5	10	b	5
25	a	6	10	b	6
15	b	8	NULL	NULL	NULL

Right outer join

③ T1 ⟗ T2

Q = B

P	Q	R	A	B	C
15	b	8	10	b	6
15	b	8	10	b	5
NULL	NULL	NULL	25	c	3

④ T1 ⟗ T2

R = C

P	Q	R	A	B	C
10	a	5	10	b	5
25	a	6	10	b	6
NULL	NULL	NULL	25	c	3

(Mixed class) — Refer notes

ADD CONSTRAINTS.

Dept

DeptID	Dept_Name
1	CS
2	IS

Emp

DeptID	EmpID	EmpName	Salary
1	101	A	10K
1	102	b	20K
2	103	C	15K

> ALTER TABLE Dept ADD CONSTRAINT PRIMARY KEY (DeptID);

> ALTER TABLE Emp ADD CONSTRAINT PRIMARY KEY (EmpID);

> ALTER TABLE Emp ADD CONSTRAINT FOREIGN KEY (DeptID) REFERENCES Dept (DeptID);

> ALTER TABLE Emp DROP CONSTRAINT EmpID;

On delete cascade:

> DELETE FROM Dept WHERE DeptID=1;

deletes all rows from Emp also with DeptID=1

Revision

student

ID	Name	Age
1	Rohit	18
2	Anita	17

1) ALTER TABLE tablename
   - ADD column_name datatype;
     eg: alter table student add marks int(3);
   - DROP COLUMN name;
     eg: ALTER TABLE student DROP COLUMN marks;
   - RENAME COLUMN oldname TO newname;
     eg: ALTER TABLE student RENAME COLUMN marks TO grades;
   - ALTER COLUMN columnname datatype;
     eg: ALTER TABLE student ALTER COLUMN grades DECIMAL (3,2);

(alter, modify, drop, rename column).

2) UPDATE tablename SET column1 = value1, column2 = value2, ... WHERE condition;
   eg: UPDATE student SET Age = 20 WHERE ID = 1;
   (update any value of record)

3) DELETE FROM tablename WHERE condition;
   eg: DELETE FROM student WHERE ID = 2;
   (delete record from table)

4) DROP TABLE table_name; (delete a table if empty)
   eg: DROP TABLE student;

5) GROUP BY
   SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s).

eg: Employee

ID	Name	City	Salary
1	Rohit	Bengaluru	20K
2	Anita	Mysore	30K
3	Alice	Mysore	40K
4	Rob	Bengaluru	25K

SELECT COUNT(ID), City FROM student GROUP BY City;

COUNT(ID)	City
2	Bengaluru
2	Mysore

Date : _____

- using two columns.

EMPLOYEE

SSN	Name	Salary	City	State
1	Alice	20000	Pune	Maharashtra
2	Bob	30000	Mysore	Karnataka
3	Charlie	25000	Bengaluru	Karnataka
4	David	40000	Bengaluru	Karnataka
5	Eve	46000	Mumbai	Maharashtra
6	Frank	45000	Mysore	Karnataka

SELECT city, State, Count(*) As Count
FROM Employee GROUP BY ~~city, State,~~ State, City;

City	State	Count
~~Pune~~	~~Maharashtra~~	1
~~Mysore~~	~~Karnataka~~	2
~~Bengaluru~~	~~Karnataka~~	2
~~Mumbai~~	~~Maha~~	

State	City	Count
Maharashtra	Pune	1
Maharashtra	Mumbai	1
Karnataka	Mysore	2
Karnataka	Bengaluru Mysore	2

6) IN - specify multiple values in WHERE clause.
7) NOTIN - " not in WHERE clause.

~~SELECT~~ SELECT * FROM tablename WHERE condition IN (value of condition);

Eg: SELECT I.D FROM Employee WHERE City IN ('Mysore', 'Pune');