- 1) OOPS concepts
- 2) Packages & Interfaces
- 3) Exception handling
- 4) Multi-Threading X
- 5) wrapper classes, Collections & File/IO Stream
- 6) Generics

Drop Box

↓

JDBC

9 - 6

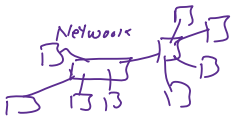
9 - 11	Concept
11 - 11.15	Break
11.15 - 1	Practical
1 - 2	Lunch
2 - 4	Concept
4 - 4.15	Break
4.15 - 6	Practical

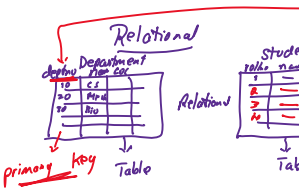
MySQL

↓

RDBMS → Relational Database Management System

DBMS
- Relational DBMS (RDBMS)
- Network DBMS (NDBMS)
- Hierarchical DBMS (HDBMS)





RDBMS (Structured)

↓

SQL

INSERT

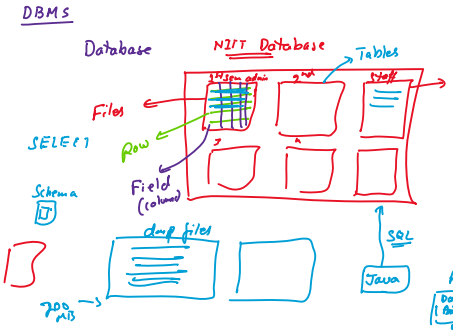
1) Oracle
2) MySQL
3) Sybase
4) H2
5) MS-Access
etc

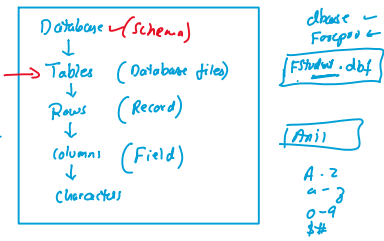
Student

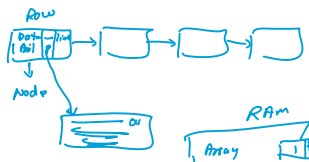
Rollno	Name	percentage
1	Anil	55.55
2	Kiran	66.66

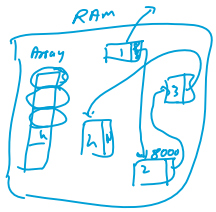
Admin : root

Password : root

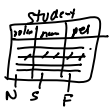


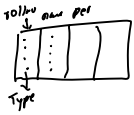






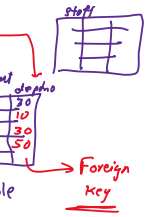
Linked List





SQL (Structured Query Language)

1) DDL - Data Definition Language → Structure of a Table → CREATE, ALTER, DROP,
2) DML → Data Manipulation Language → INSERT, UPDATE, DELETE
3) DQL → Data Query Language → SELECT
4) DCL → Data Control Language → GRANT, REVOKE
5) TCL → Transaction Control Language → COMMIT, ROLLBACK



Un-structured

1) MongoDB → NoSQL

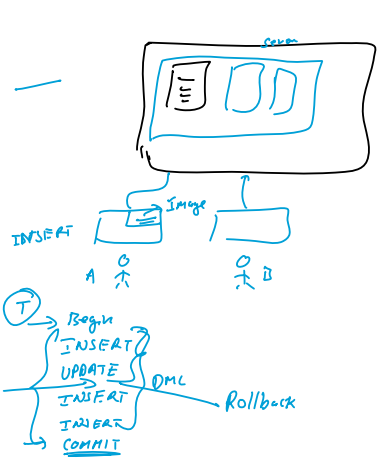
Collection (Student)

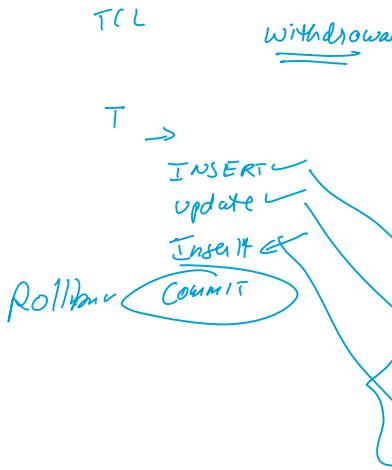
[{rollno: 1001, name: "Anil", percentage: 55.55},
{rollno: 1002, name: "Ravi", marks1: 55, marks2: 60}]

Data Structures

Linked list

Tree

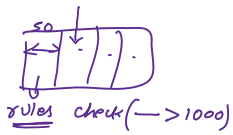




DDL (Data Definition Language)

1) CREATE
2) ALTER
3) DROP

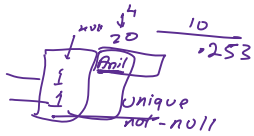
CREATE




```
CREATE TABLE table_name
(
    column_name1   Datatype(width)   Constraints,
    column_name2   Datatype(width)   Constraint,
    :
);
```

SQL

0000
9999




```
CREATE TABLE student
(
    rollno number(4) primary key,
    name VARCHAR(100) NOT NULL,
    percentage number(10,2) CHECK(percentage >= 0),
    email varchar(100) UNIQUE NOT NULL
);
```

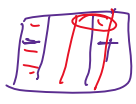
ALTER

```
ALTER TABLE table_name
[ ADD | MODIFY | DROP ]  Columnname  Datatype(width)  constraint
```

```
ALTER TABLE student
    ADD phone varchar(15);
```

```
ALTER TABLE student
    MODIFY name CHAR(75);
```

```
ALTER TABLE student
    DROP email;
```

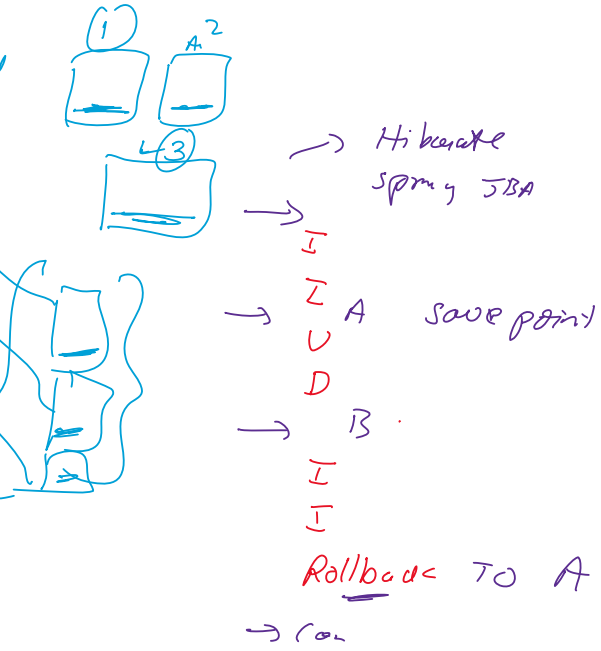


DROP

```
DROP TABLE table_name
```

Student /

DML DDL



→ Hibernate
spring JBA

→ I
I
U
D
I
I

A save point

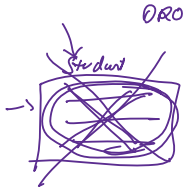
→ B .

Rollback TO A

→ con

DROP

DROP TABLE table-name



DROP TABLE student

DML DDL

TRUNCATE →

DDL ← TRUNCATE TRUNC

TRUNCATE TABLE student;

DML → INSERT, UPDATE, DELETE

INSERT

```
INSERT INTO table-name (column-names)
VALUES(column-values)
```

a) INSERT INTO student
VALUES (1, 'Anil', 55.55)

b) INSERT INTO student
VALUES (1, 'Anil', null)

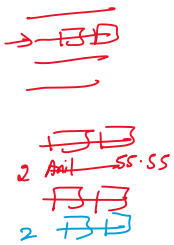
c) INSERT INTO student(studentid, studentname)
VALUE (1, 'Anil')

d) INSERT INTO student (studentname, studentid)
VALUE ('Anil', 1)

e) INSERT INTO studentbackup SELECT * FROM student;

UPDATE

```
UPDATE table-name
SET column1 = value1,
    column2 = value2,
    :
WHERE condition;
```



e.g

```
UPDATE student
SET studentname = 'Anil Kumar',
    percentage = 66.77
WHERE studentid = 2;
```

1	Kiran	55.55
2	Ravi	66.66
3	Rakesh	77.77
4	Prateek	88.88

DELETE

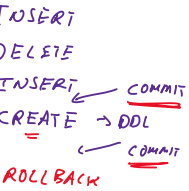
```
DELETE FROM table-name
WHERE condition;
```

eg DELETE FROM student
WHERE rollno > 2;

'ANIL'

DELETE FROM student
WHERE studentname = 'Anil';

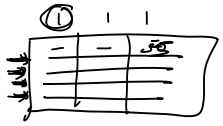
DELETE FROM student
WHERE rollno > 10 and studentname = 'Anil';


INSERT
DELETE
INSERT
COMMIT
CREATE
DDL
COMMIT
ROLLBACK

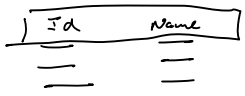
```
DELETE FROM student
WHERE rollno >=10 and rollno <=20 ;
```

SELECT

```
SELECT * | Column-names
FROM table-name
WHERE condition
ORDER BY Column-name
```

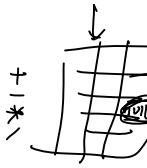


a)
```
SELECT *
FROM student
```



b)
```
SELECT studentid, studentname
FROM student;
```

c)
```
SELECT studentid AS ID, studentname AS Name
FROM student
```



d)
```
SELECT studentId, studentname, percentage + (percentage * 0.10) As Percentage
FROM student
```

e)
```
SELECT studentid, studentname, percentage, NVL(percentage, 0)+(NVL(percentage
FROM student;
```

f)
```
SELECT *
FROM student
WHERE studentid = 5;
```

g)
```
SELECT *
FROM student
WHERE percentage > 70
```

h)
```
SELECT studentid AS ID, studentname AS Name
FROM student
WHERE percentage > 70
```

5, 8, 13, 25

i)
```
SELECT *
FROM student
WHERE studentid = 5 OR studentid = 8 OR studentid
```

⟶

```
SELECT *
FROM student
WHERE studentid IN (5, 8, 13, 25);
```

j)
```
SELECT *
FROM student
WHERE percentage >= 60 AND percentage <=80;
```

⟶

```
SELECT *
FROM student
WHERE percentage BETWEEN 60 AND 80
```

e

,0) * 0.10) As Inc_Percentage

> < >= <= != == → Logical operators

SQL operators

1) IN
2) BETWEEN .. AND
3) LIKE

wild cards

% → Any no of chars & All chars
_ (underscore) → Single any char

= 13 OR Studentid 25

k) SELECT *
FROM student
WHERE studentname Like 'A%'

l) SELECT *
FROM student
WHERE studentname Like '_A%';

m) SELECT *
FROM student
WHERE studentname Like '%THN';

ORDER BY → sorting

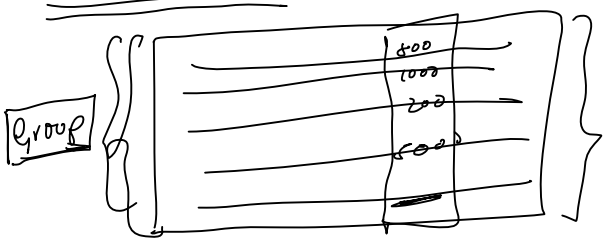
a) SELECT *
FROM student
ORDER BY studentname;

b) SELECT *
FROM student
ORDER BY studentname DESC

c) SELECT *
FROM student
ORDER BY percentage DESC, name

80
60
50

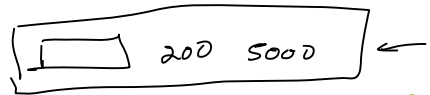
GROUP By clause



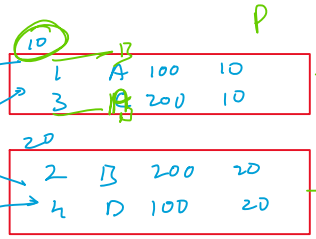
Sum
Count
Min
MAX
AVG

Emp
empno name sal

SELECT sum(sal), min(sal), max(sal)
FROM emp

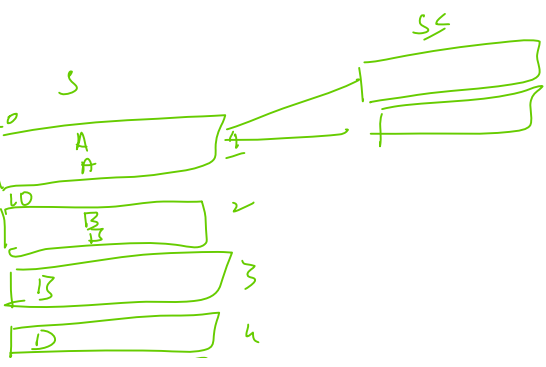


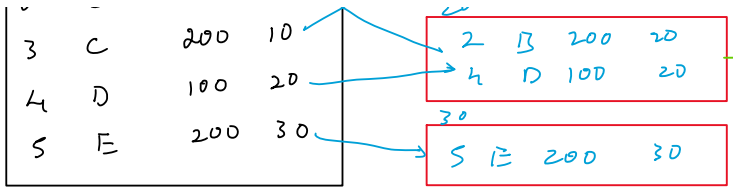
1	A	100	10
2	B	200	20
3	C	200	10
4	D	100	20



Rakesh
Anil
Kiran

80 Anil
80 Kiran
80 Rakesh
70 Arun
70 Ravi

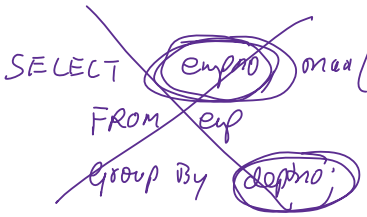


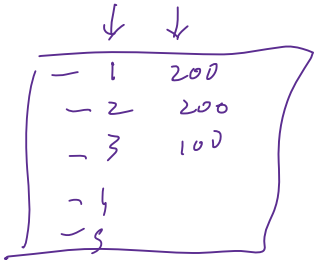


GROUP BY

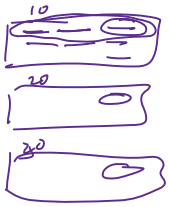
```
SELECT  max(sal), deptno
  FROM  emp
  GROUP BY  deptno, name, sal
```

200
200
200





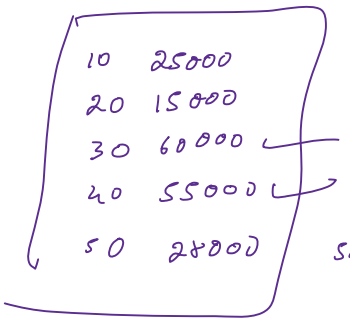
```
SELECT  deptno, max(sal)
  FROM  emp
  Group By  deptno;
```

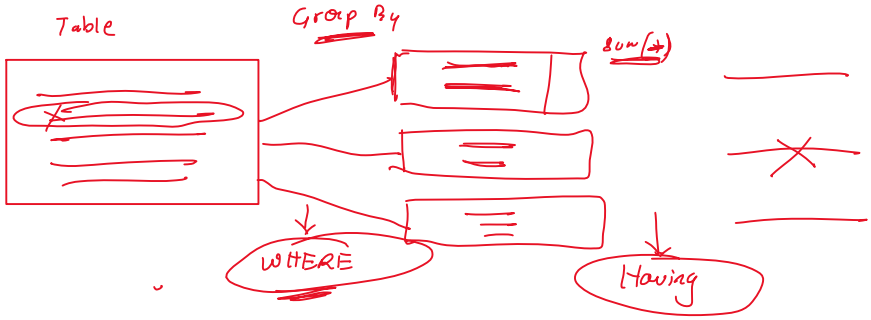


Having

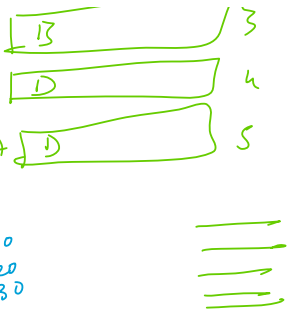
```
SELECT  deptno, max(sal)
  FROM  emp

  Group By deptno
  HAVING  max(sal) > 50000;
```

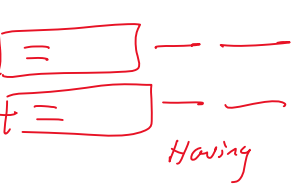
deptno max(sal)
Group deptno






```
SELECT  deptno, Sum(sal)
  FROM  emp
  WHERE  deptno != 30
  Group By deptno
  HAVING  max(sal) > 50000;
```


3
4
5


Having

```
WHERE deptno != 30
Group by deptno
HAVING Sum(sal) > 50000;
```

where Group

Joins

3 X 4 = 12

Dept

deptno	dname
10	CS
20	MECH
30	IS

Emp

empno	ename	deptno
1	A	20
2	B	10
3	C	20
4	D	10

empno ename deptno dname

```
SELECT emp.empno, emp.ename, emp.deptno, dept.deptno, dept.
    FROM emp, dept
    WHERE emp.deptno = dept.deptno;
```

- Equi Join

Non-Equi-Join

Emp

1	A	500
2	B	800
3	C	200
4	D	900

Salgrade

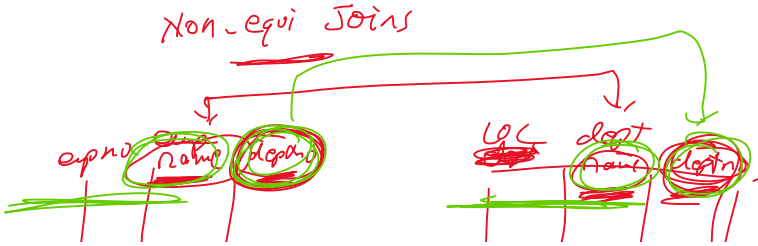
grade	minsal	maxsal
A	501	1000
B	301	500
C	0	300

1	A	500	A
1	A	500	B
1	A	500	C
2	B	800	A
2	B	800	B
2	B	800	C

empno	name	sal	grade
1	A	500	B
2	B	800	A
3	C	200	C
4	D	900	A

```
SELECT emp.empno, emp.ename, emp.sal, salgrade.grade, salgrade.min
    FROM emp, Salgrade
    WHERE emp.sal >= Salgrade.minsal AND
          emp.sal <= Salgrade.maxsal;
```

Non_equi Joins



Having

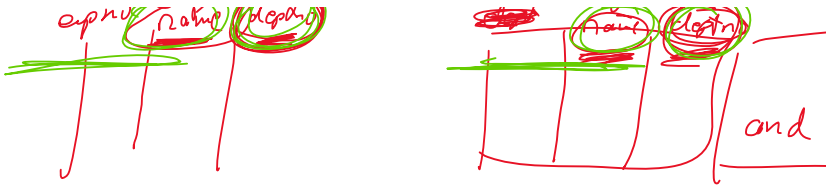
1	A	20	10	CS
1	A	20	20	Mech
1	A	20	30	IS
2	B	10	10	CS
2	B	10	20	Mech
2	B	10	30	IS
3	C	20	10	CS
3	C	20	20	Mech
3	C	20	30	IS
4	D	10	10	CS
4	D	10	20	Mech
4	D	10	30	IS

dname

501	1000 X
301	500
0	300 X
501	1000
301	500 X
0	300 X

sal, salgrade. maxsal

empdeptno = dept dn




```
SELECT empno, ename, dname, d
  FROM emp
  natural join dept;
```

emp.deptno = dept.dn

emp.name = dept.nam

deptno