Date ___/___/_____

TINYINT (-128 to 127)

UNSIGNED TINYINT → (0 → 255)

Advanced data type.

1) JSON

data transfer .x

key, value pair (JS)

SQL Types of Commands in DDL

defining relational schema.

1) CREATE *
2) ALTER TABLE
3) DROP → delete table DB, view
4) TRUNCATE : remove all tuple from the table.
5) RENAME

DRL / DQL Retrieve, Query

1) SELECT

DML

1) INSERT
2) UPDATE
3) DELETE

4) DCL

GRANT access privilege

REVOKE. revoke user access privilege (remove) grant

5) TCL TRASACTION CONTROL LANGUAGE

START

COMMIT Apply changes & end

in when ROLLBACK discard changes & end

SAVEPOINT checkout write group of transaction

Page No.

DROP DATABASE IF EXISTS temp;
// deletes whole database

DATA RETRIEVAL

RIGHT → LEFT.

Order of execution

SELECT * FROM table_name;

SELECT 44+12 O/P - grid
dummy table.

select now();
select lcase("SHRIDIP") → swaam
ucase(shridip) → SHRIDIP

SELECT * FROM WORKER WHERE SALARY > 100000;
WHERE dept = 'IT';
+ attribute
WHERE SAL BETWEEN 80 AND 300;
INCLUSIVE →

- reduce OR statement
HR / ADMIN
SELECT * FROM WORKER dept = 'HR' OR dept = 'Admin'
HR / ADMIN / ACCOUNT
multiple OR

Bench way IN keyword

SELECT * FROM where dept IN 'ACCOUNT', 'HR', 'ADMIN';

NOT (IN)
NOT IN
AND OR NOT

IS NULL

WHERE Status IS NULL

('Rohan', 18, NULL) ✓
('Kirti', 18, placed)

(a) WILDCARD / pattern learning
(%, *, _)

(%p_)
Second last (p) ✓

_pa_ replaces
apaa

where first_name LIKE '%i'; → any number of word
'_i'; → one word

Date

Sorting -

SELECT * from worker ORDER BY {depart} ← ASC

SELECT * FROM WORKER ORDER BY SALARY DESC

DISTINCT

HR
HR
Admin
Admin
Account

HR
Admin
Account

SELECT ~~DISTINCT~~ DEPARTMENT

SELECT DISTINCT DEPARTMENT FROM WORKER

Data Grouping

(HR → count
Admin
~~Account~~

Amazon company → India
USA

Aggregation
group

GROUP BY AGGREGATION FUNCTION

COUNT (*)

SELECT DEPARTMENT FROM Worker GROUP BY DEPARTMENT

↓

always aliasing

aggregation functions X

Page No.

GROUP BY - DEFAULT (COUNT)

Date ___/___/___

country wise →

dept →

dep department avg salary

SELECT DEPARTMENT, AVG(SALARY)
FROM WORKER
GROUP BY DEPARTMENT;

MIN(SALARY)

INTERNALLY CODEBASE.

(OUNT)

MIN

SELECT → WHERE
filter

GROUPBY → HAVING

CLAUSE
↓
filter

AGGREGATION.

similar.

condition

→ department crore having more than 2 worker

grouping having

COUNT(DEPARTMENT)

SELECT DEPARTMENT FROM WORKERS

GROUP BY DEPARTMENT HAVING COUNT(DEPARTMENT) > 1;

("\\ backslash");
" \"
(\n newline);
("\t tabe";

SQL →

CONSTRAINTS –

PRIMARY KEY
not null
unique
one one PK.
PK int (best practice

② FOREIGN KEY ;
refer to primary key.

PRIMARY KEY (id);

refers PK of other table.

Date __/__/____

Saathi

CREATE TABLE customer (

id integer PRIMARY KEY,
name varchar(255);
adress varchar(255);
gender char(2);
city varchar(255)
Pincode integer
);

CREATE TABLE order_details (

order-id integer PRIMARY KEY,
delivery date DATE;
cust-id INT
FOREIGN KEY (cust-id) references customer(id);
↓
Tablename

(3) UNIQUE
create table customer (
username VARCHAR(255)

(4) CHECK
consistency contraints
acc balance > 1000.
error

(5) DEFAULT
balance INT NOT NULL

amazon prime not prime

Attribute PK & FK

Date ___/___/_____

Saath

ALTER OPERATIONS

ALTER TABLE account
ADD interest FLOAT NOT NULL DEFAULT 0;

MODIFY ->

→

SWITCH ENHANCED

switch(x)

case X •

~~case X 1~~

case 1 -> {
sout(Hello)
sout(Greetings)
}.

no need of breaks.

sort in order of ~~finishing~~ start time

1,2,1
3,4,3

8th Purple

ALTER TABLE

Date __/__/__

prime_status not null default false.

ALTER OPERATIONS →

ADD NEW COLUMN

ALTER TABLE account ADD interest FLOAT NOT NULL 0;

MODIFY

ALTER TABLE account MODIFY interest DOUBLE NOT NULL 0;

DESC ACCOUNT;
describes account

CHANGE COLUMN

→ (old_name, new_name, datatype)

ALTER TABLE account CHANGE COLUMN

interest saving_interest DOUBLE NOT NULL 0;

DROP COLUMN

ALTER TABLE account ~~rename~~ DROP column

saving_interest;

RENAME COLUMN - TABLE-

ALTER TABLE account RENAME TO account_detail

DML —

Saathi

Date __/__/____

INSERT, ~~INTO~~

UPDATE,

insert into Customer values (id, cname).

SET SQL_SAFE_UPDATES = 0;

UPDATE INTO Customer

where id =

delete

↓

ON DELETE Cascade Parent/Child =

ON DELETE SET NULL Child = NULL

REPLACE →

① present → replace REPLACE

② not present → insert INSERT.

REPLACE INTO ~~FA~~ CUSTOMER

UPDATE → present → update

not present → do nothing

JOINS

15/12/22

Date

for Foreign key used to reference one table to other

Saathi

Foreign Key.

→ Data fetch → JOINS.

Parent — LEFT Table (PK) (FK) RIGHT Table — Child

Key	Key
1	1
2	2
3	3
4	X

1. INNER JOIN —

→ COMMON (MATCHING DATA SHOWN)

$$\begin{matrix}1\\2\\3\end{matrix} \times \begin{matrix}1\\2\\3\end{matrix}$$

To apply join there should be common attributes

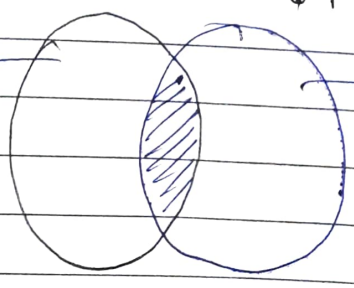
SYNTAX

```
Select C.*, O.* FROM customer AS C
INNER JOIN orders as O
ON cid = O.cid
```

objectname — C., O.

aliasing — AS C — temp name for particular query

Left ← → Right



INNER JOIN

VENN DIAGRAM

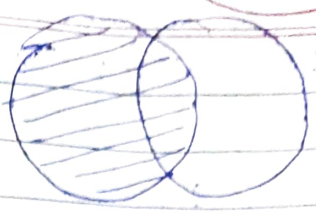
Page No.

OUTER JOIN

Saathi

Date __/__/____

OUTER JOIN.



LEFT JOIN.

left → all
right → matching (left & right)

L1	L2	key	key	R1	R2
		1	1		
		2	2		
		3			
		4			

Result

key	L1	L2	R1	R2
1.	←	→	matching	
2.	←	→		
3.	←	→	NULL	null
4.	←	→	null	null

SYNTAX

```
SELECT C.* , O.* FROM [CUSTOMER] [AS | C]
LEFT JOIN [orders] [AS | O]
ON C.id = O.cust_id;
```

Page No.

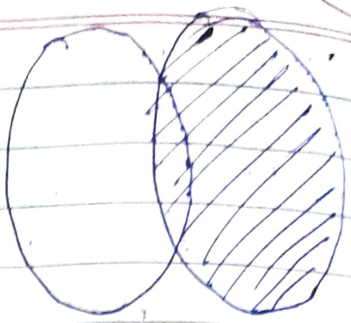
Left Table only matching / Considers Right Table

Concerned about Rig~~ht~~

Date ___/___/_____

Saathi

(3) RIGHT JOIN —



Left Table

L1, L2 key	key	R1	R2
1			
2	2		
3	3		
	4		
	5		

SYNTAX

SELECT C.*, O.* FROM Customers AS C, ~~Orders~~ RIGHT JOIN, orders as O, ON customer_id = order_id

Result

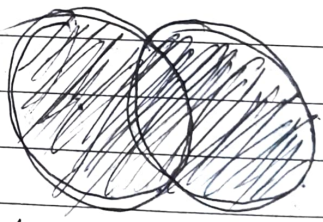
key	L1	L2	R1	R2
1	←	→		
2		→		
3	←	→		
4	null		←	→
5	null		←	→

Matching

(4) FULL JOIN — (LEFT JOIN + RIGHT JOIN

No keyword emulate =

LEFT JOIN
RIGHT JOIN
UNION OF BOTH



NOTE **

ON (KEY) KEYWORD

MYSQL (FULL JOIN) ✗ NO KEYWORD KEYWORD

Emulated (Left Join ∪ Right Join)

Select * from lefttable as l ~~LEFT JOI~~
LEFT JOIN Righttable AS R
ON l.key = r.key. (key - cid, oid)
UNION

Select * FROM LEFTABLE as l
~~LEFT~~ RIGHT JOIN Righttable as R
ON key = l.key

Alias - A temp name for a particular query.

Date / /

Saathi

~~SELECT * FROM TABLEEETTABLE~~

SELECT * FROM TABLE1 as t1
LEFT JOIN TABLE2 as t2
ON t1.key = t2.key

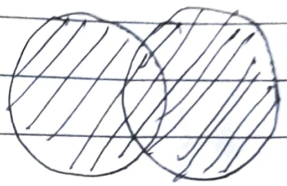
As
→ Alias
temp name for a query)

key = cid.
key = oid

UNION.

SELECT * FROM TABLE1 as t1
RIGHT JOIN TABLE2 as t2
ON t1.key = t2.key

LEFT JOIN.
UNION
RIGHT JOIN (FULL JOIN)
= FULL JOIN



Result.

Left			Right		
L1	L2	key	key	R1	R2
		1	1		→ matching
		2	2		→
		4		null	null
		5		null	null
null	null		6		→
null	null		7	←	→

Page No.

Date __/__/____

Saath

5. CROSS JOIN → Cartesian Product

Makes Keys like cid
rid

→ Here are requirement

Simple multiply

NO INDUSTRIAL USE

L	L2
1	A
2	B

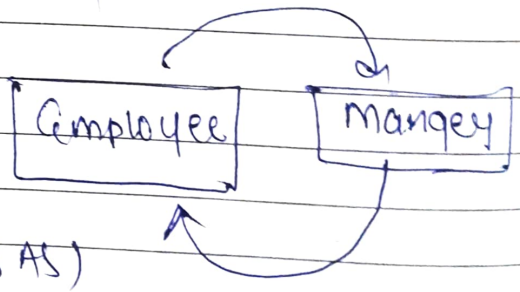
R1	R2
3	X
4	Y

IL RR
5 rows 10 rows

Result

L1	L2	R1	R2
1	A	3	X
1	A	4	Y
2	B	3	X
2	B	4	Y

6. SELF JOIN—



→ emulate (INNER JOIN)
only common ← Alias AS)
→ NO KEYWORD

Beware of Buna phe-

```
select e1.id, e2.id, e2.name
FROM employee as e1 → LEFT TABLE
    INNER JOIN.
    employee as e2 → RIGHT TABLE
ON e1.id = e2.id.
```

Page No.

Date __/__/____

Double hyphen is a comment in MySQL

Saathi

Project

Employee.

Client.

Q Enlist all employees ID's, names along with Project allocated & client (Inner join)

Employee + client

~~select~~

SELECT e.id, e.name ~~FROM~~
FROM employee ~~AS~~ e
INNER JOIN
Projects AS p
ON p.id = p.empID // should be same key (PK = FK)

-- this is a comment

Q Fetch out all the employee ID's ~~not~~ and their contact details who have been working from Jaipur with the clients name working in Hyderabad.

SELECT E.id, E.name, E.mail, E.phone, ~~E.~~
C.name as E
INNER JOIN Client as C
ON ~~Client~~ E.id = C.empID
WHERE E.city = 'Jaipur' AND
C.city = 'Hyderabad';

Page No.

6 HOURS → sleep = 12 hours → 12 + 3 hours = 15 hours
8 HOURS →
12 HOURS → STAY ACTIVE → 12 HOURS + 6 hours = 18 hours

Date

-- LEFT JOIN

fetch out each project allocated to each employee

```
select * from employee as E
LEFT
FROM E
  LEFT JOIN Project as P
    ON  p.id = E.empId.
```

-- RIGHT JOIN --

List out all the projects along with the employee name and their respective allocated emp ID.

```
p.name, e.name
SELECT p.id FROM projects as P
  RIGHT JOIN.        employee as
  employees as E project as p
  ON  e.id = p.empid
```

CAN WE USE INNER JOIN WITHOUT JOIN KEYWORD

```
Select * from employee as E
mixed
  project as P
    where
    e.Id. pid = p.emID;
SELECT * FROM EmployeeAsE,
project as P,
WHERE  e.id = p.empId
```

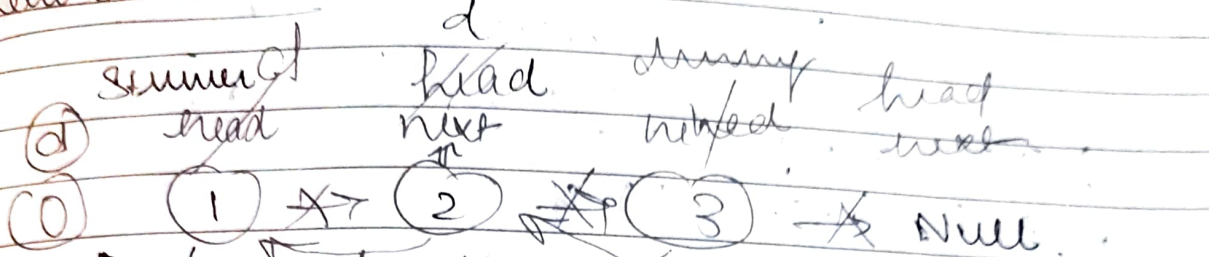
Page No.

Date __/__/____

Saathi

SET OPERATIONS → & SUB QUERIES →

Reverse a LL



head.next = dummy

dummy = dummy.next

head = head.next

return dummy

ListNode dummy = new ListNode

while (head != null)

ListNode next = new ListNode

Next = head.next

dummy. = head. next =

head = head.next

dummy = head;

head = head.next

SET OPERATION -

Date ___ / ___ / ____

Saatr

Sobbonash A

CROSS JOIN -

```
select * from customer as c
CROSS JOIN projects as p;

select * from lefttable as l, right table as r
                    where l.id = r.id
```

SET

~~UNO~~.

- UNION
- INTERSECTION
- MINUS

* Used to combine multiple
1. select statements
2. Always gives distinct rows

JOIN	SET Operations
Column wise combination	Row wise combination
Combines tables based on matching conditions	data types columns from each table should be same
distinct / duplicate rows	distinct rows.
number of cols may not be same	no. of cols. must be same.
Combine results horizontally	Combines results vertically

INTERSECTION — DISTINCT, INNER JOIN

MINUS — LEFT JOIN

Saathi

NION

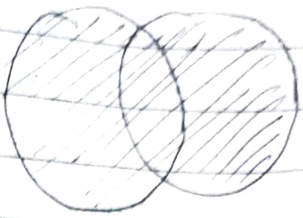
ombiny two or more select statements

NTAX

SELECT * FROM TABLE1

UNION

SELECT * FROM TABLE2



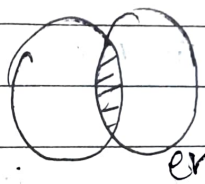
. No. of cols, order must be same for TABLE 1 & TABLE 2

INTERSECTION — NO KEYWORD (INNER JOIN) emulate

SELECT * from T1

INTERSECTION X

Select * from T2;



emulate

```
Select DISTINCT id from T1
INNER JOIN T2
        USING (id);
```

MINUS X No Keyword

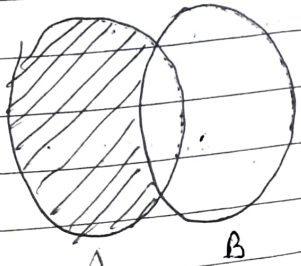
EMULATE

SELECT ~~* from~~ T1

LEFT JOIN T2 USING (ID);

WHERE T2.id IS NULL

A-B



A B

Date ___/___/_____

Saath...

Questions on set operations

Dept 1			Dept 2		
empid	name	role	empid	name	role

--UNION

```
SELECT * FROM dept1
UNION
dept2
SELECT * FROM dept2
```

DISTINCT Rows.

~~SELECT * FROM dep.~~

SELECT DISTINCT role from
dept1, dept2
~~WHERE~~ ~~US~~
WHERE ~~dept~~ role
= "Salesman";

SELECT * FROM dept1 WHERE role
UNION. = Salesmen
SELECT * FROM dept1 where role = Salesman

dept.* →

SELECT * FROM dept1
INNER JOIN.
dept2 USING (empid);

-- list all the employees working in dept1 ~~and~~ not not dept2

SELECT * from dept1
LEFT JOIN — dept2 USING(

Page No.

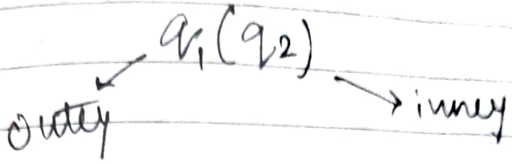
SUB QUERIES →

Date __/__/____ TO

saathi

ALTERNATE ~~FOR~~ JOINS

OUTER QUERY
INNER QUERY ↘ depends ←



select * from ~~table~~ name where id
IN (select id from table
where name = '~~Lak~~ Shri')

3. Nested Query

PROJECT, EMPLOYEE, CLIENTS.

WHERE clause same table
(employees with age > 30).
→ select * from employee where age IN (select age from employee where age > 30)

WHERE clause different table
employee details. working in more than 1 project

select * from employee where id IN (~~XXX~~

Select empID from project GROUP BY empID
having COUNT (empID) > 1);

Employee as E
Employee E
Employee e

Date __/__/____

Saathi

Single value subquery

emp details having age > avg(age)

select * from employee where age > (select avg(age) from Employee;

~~employee~~

FROM clause → derived table (need Aliasing)

Select max age person whose first name contains 'a'

select * from ( )

or

Select max(age) FROM, → returns a table

~~where name~~

(Select * from Employee

WHERE FirstName LIKE '%a%'

) AS temp;

** CORELATED SUBQUERY depends → need Aliasing

depends (outer / inner) depends

outer (inner)

depends

3rd oldest Employee →

```
SELECT * FROM Employee e1.
WHERE 3 = (
SELECT COUNT (e2.age)
FROM EMPloyee e2.
  WHERE e2.age >= e1.age
);
```

Page No.

Date (WHERE)

Saathi

① e1.age = 32 → ② ✗

② e1.age = 44 → ③ ✓ included.

③ 22 → ✗

④ 31 → ✗

32
44
22
31
21

for each outer query inner query will run

→ SYNTAX

```
SELECT col1, col2
FROM table1 as outer ←
WHERE col1 [operator] (

  SELECT col1, col2
  FROM table2
  WHERE exp1 = outer.exp2 )
```

Alias, object

inner query driven by outer query ↑ in other words

Correlated subquery executes once for each candidate row considered by the outer query

CREATE VIEW show AS
SELECT name, age from Student;

Date

JOINS	Vs	SUBQUERY
• faster		• slow
• max burden on DBMS		• calc based on query by user
• complex, implementation difficult		• comparatively easy and implement
• Choosing OPTIMAL JOIN FOR BIG DATA IS DIFFICULT.		• EASY

Select * from Student where age > 22

SQL VIEWS →

MySQL → views.

customer → id | name | age | address

view → name | age → alias

custom view

only shows defined attributes in view.

Q) question

-- VIEW
Select * FROM Employee

→ CREATE VIEW employee-view AS.
SELECT name, age FROM EMPLOYEE

-- VIEW THE VIEW
SELECT * FROM VIEW;

VIEW	
Yash	22
Shridep	21

-- ALTER VIEW

→ ALTER VIEW employee_view AS
SELECT name FROM EMPLOYEE

→ DROP VIEW IF EXIST employee_view

name

Yash
Shridep

→ error if not exist

Page No.