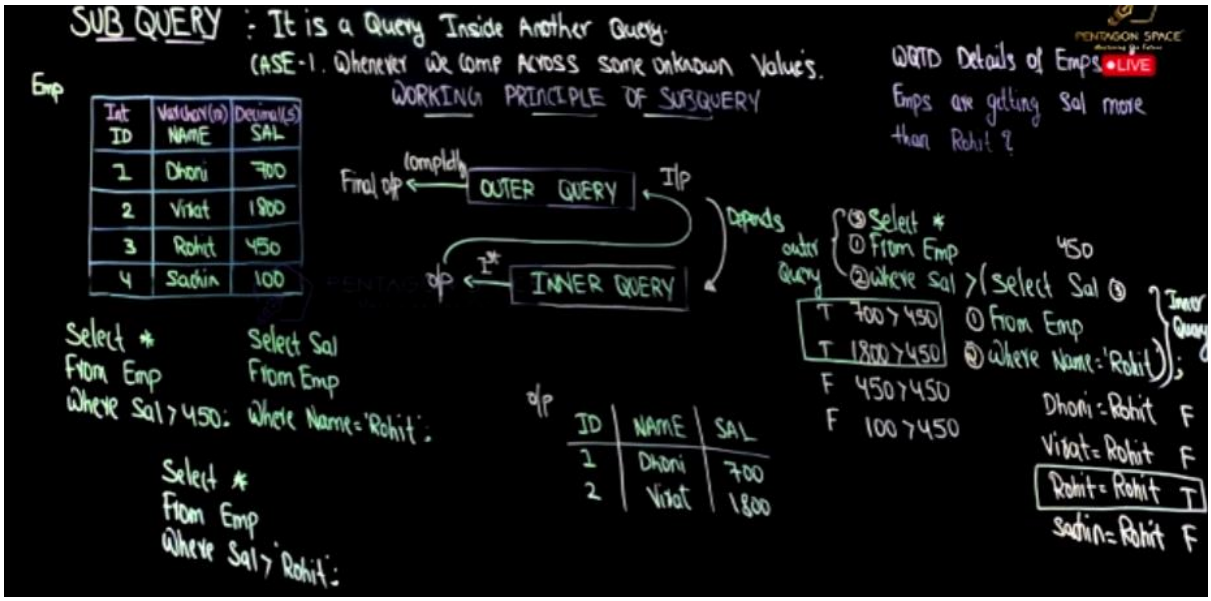
SQL Session-2

14/07/25 – Monday

Sub Query:

It is query inside another query

CASE-1:



Working Principles of Sub-Query

- In sub-query inner query will executes first and generates output
- Output of inner query will be given as input to the outer query
- Outer query executes completely after te inner query by taking input from inner query
- Outer query depends on inner query
- Inner query doesn't depends on outer query

Rules of Sub-Query

1. The column_name selected in inner query and te column_name written inside outer query should be of same datatype

```
WHERE SAL>(SELECT SAL          VALID
      INT         INT

WHERE SAL>(SELECT COMM         VALID
      INT         INT

WHERE SAL>(SELECT FNAME        INVALID
      INT         VARCHAR(10)
```

2. We can select only one column inside inner column

```
EX:

WHERE SAL>(SELECT SAL,COMM      INVALID

WHERE SAL>(SELECT SAL           VALID
```

Wqtd details of emps who are getting are getting salary less than Priya

```
select *
   -> from emps
   -> where sal < ( select sal
   -> from emps
   -> where fname= 'priya');
```

Wqtd fname, lastname, job and sal if emps are working same as murli's job role

```
select fname,lname,job,sal
   -> from emps
   -> where fname!='murali' and  job = ( select job
   -> from emps where fname = 'murali');
```

Wqtd fname and lname together as full name along with salary,job and lid if emps are working in the location same as suresh location and getting salary more than Priya

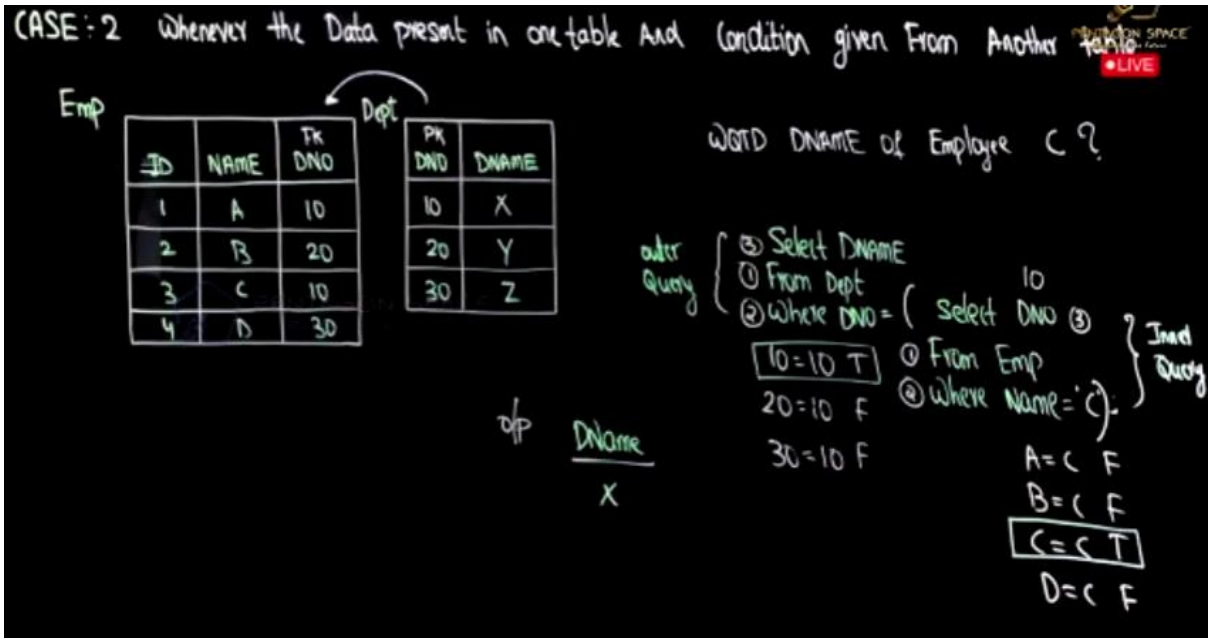
```
select concat(fname,' ',lname) as full_name,sal,job,lid
   -> from emps
   -> where lid=(select lid
   -> from emps
   -> where fname='suresh') and sal>(select sal
   -> from emps
   -> where fname = 'priya');
```

15/07/25 - Tuesday

Sub query:

CASE-2:

Whenever the data present in one table and condition given from another table.



Wqtd city name of the employee kiran

```
select city
    -> from locations
    -> where lid = (select lid
    -> from emps
    -> where fname = 'kiran');
```

Wqtd details of emps who is living in the state rajasthan

```
select *
    -> from emps
    -> where lid = ( select lid
    -> from locations
    -> where state = 'rajasthan');
```

Select city and state of the customer rohit sharma

```
mysql> select city,state
-> from locations
-> where lid = (select lid
-> from customers
-> where name = 'rohit sharma');
```

Wqtd item details which belongs to spice hub restaurant

```
mysql> select *
-> from menu_items
-> where restaurant_id = (select restaurant_id
-> from restaurants
-> where name= "spice hub");
```

17/07/25 – Thursday

Wqtd details of employees who are working as security or manager in Mumbai city

```
select *
-> from emps
-> where job in ('security','manager') and  lid=(select lid
-> from locations
-> where city = 'mumbai');
```

Wqtd details of the customers who are living in Chennai or Jaipur city

```
select *
-> from customers
-> where lid = ( select lid
-> from locations
-> where city in ('chennai','jaipur'));
```

ERROR 1242 (21000): Subquery returns more than 1 row

Types of Sub query

1. Single row subquery
2. Multi row subquery
3. Co-related subquery

1. **Single row subquery:**
   - If inner query return a single value we can consider that query as single row subquery.
   - Here we can use both normal operators (=, !=, >, <) and special operators (in, not in, all, any)
2. **Multi row subquery:**
   - If inner query returns more than one value, we can consider that query as multi row subquery
   - Here we can use special operators (in, not in, all, any)

Wqtd details of the customers who ordered some product

```
select *
    -> from customers
    -> where order_id is not null;
```

Or

```
select *
    -> from customers
    -> where order_id in (select order_id from orders);
```

Wqtd details of emps who delivered the product to the customer who belongs to Kolkata city

```
mysql> SELECT *
    -> FROM EMPS
    -> WHERE EID IN(SELECT EID
    ->              FROM ORDERS
    -> WHERE STATUS='DELIVERED' AND ORDER_ID IN(SELECT ORDER_ID
    ->                                          FROM CUSTOMERS
    ->                                          WHERE LID IN(SELECT LID
    ->                                                       FROM LOCATIONS
    ->                                                       WHERE CITY='KOLKATA')));
```

Wqtd name of the customers whose payment status is failed

```
select name
    -> from customers
    -> where order_id in (select order_id
    -> from orders
    -> where order_id in (select order_id
    -> from payments
    -> where status = 'failed'));
```

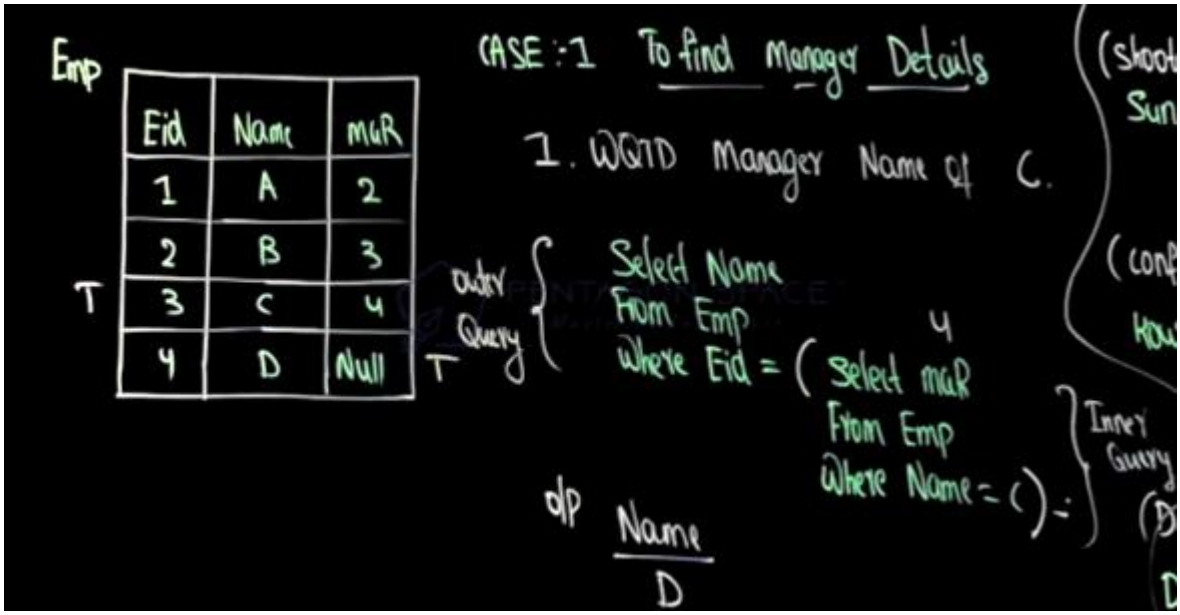
Wqtd restaurants name which never received a review

```
select name
    -> from restaurants
    -> where restaurant_id not in (select restaurant_id
    -> from reviews);
```

18/7/25- Friday

Employee and Manager Relationship

Case 1:



Wqtd manager details of Suresh

```
select *
    -> from emps
    -> where eid = (select mgr
    -> from emps
    -> where fname="suresh");
```

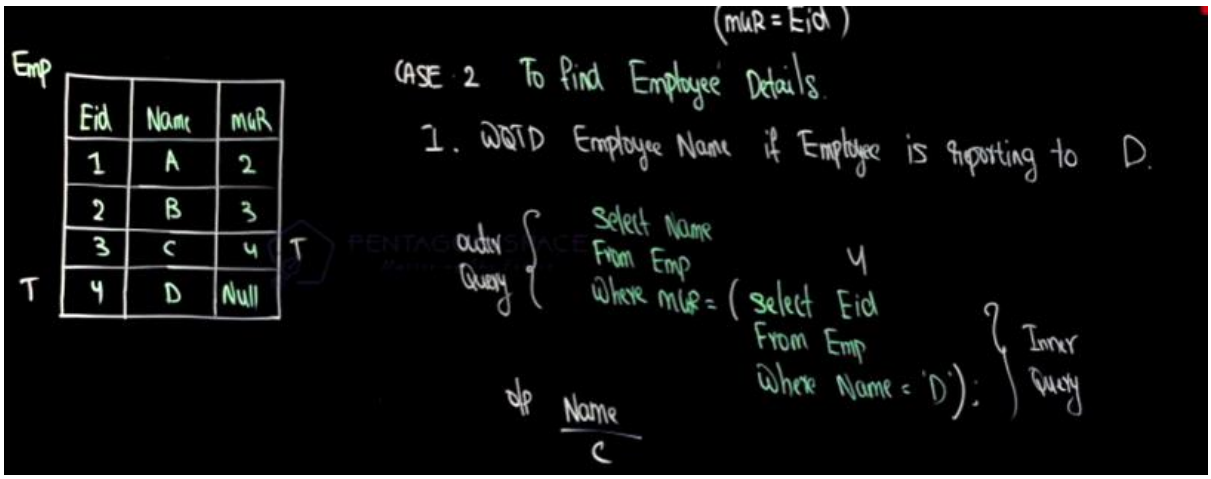
Wqtd city name of divya’s manager

```
select city
    -> from locations
    -> where lid in (select lid
    -> from emps
    -> where eid in (select mgr
    -> from emps
    -> where fname = 'divya'));
```

Wqtd details of aman’s managers manager

```
select  *
    -> from emps
    -> where eid in (select mgr
    -> from emps
    -> where eid in (select mgr
    -> from emps
    -> where fname = 'aman'));
```

Case -2 :



Wqtd details of emps who are reporting to Jahnavi

```
select *
    -> from emps
    -> where mgr in (select eid
    -> from emps
    -> where fname = 'jahnavi');
```

Wqtd details of emps who are reporting to arjun’s manager

```
select *
    -> from emps
    -> where mgr in (select eid
    -> from emps
    -> where eid in (select mgr
    -> from emps
    -> where fname = 'arjun'));
```

Wqtd location details of the emps who are reporting to faizans manager’s manager

```
select *
    -> from locations
    -> where lid in (select lid
    -> from emps
    -> where mgr in (select eid
```

-> from emps

-> where eid in (select mgr

-> from emps

-> where eid in (select mgr

-> from emps

-> where fname = 'faizan'))));

21/7/35-Monday

All

- It is a multi value operator which takes multiple values at the rhs and single at the lhs along with relational operators.
- **Syntax:**

```
          LHS                                    RHS
COLUMN_NAME/EXPRESSION </>/<=/>= ALL(V1,V2,.....VN);
```

- It works on and condition

Wqtd details of the emps who are getting salary less than waiters

select *

-> from emps

-> where sal<all(select sal

-> from emps

-> where job='waiter');

Any

- It is a multi value operator which takes multiple values at the rhs and single value at the lhs along with relational operators
- **Syntax:**

```
          LHS                                    RHS
COLUMN_NAME/EXPRESSION </>/<=/>= ANY(V1,V2,.....VN);
```

- It works on or condition

Wqtd fname of the emps if employee is reporting to jahnavi and getting salary less than murali and living in the state Karnataka

```
select fname
   -> from emps
   -> where mgr in(select eid
   -> from emps
   -> where fname= 'jahnavi') and sal < (select sal
   -> from emps
   -> where fname='murali') and lid in (select lid
   -> from locations
   -> where state = 'karnataka');
```

Drawback of subquery:

Here we can't retrieve the data from multiple tables simultaneously

---

Joins

It is used to retrieve the data from multiple tables simultaneously

Types of joins

1. Cross Join/Cartesian join
2. Inner join
3. Outer join
   i. Self outer join/left join
   ii. Right outer join/ right join
4. Self join
5. Natural join

1. **Cross join:**

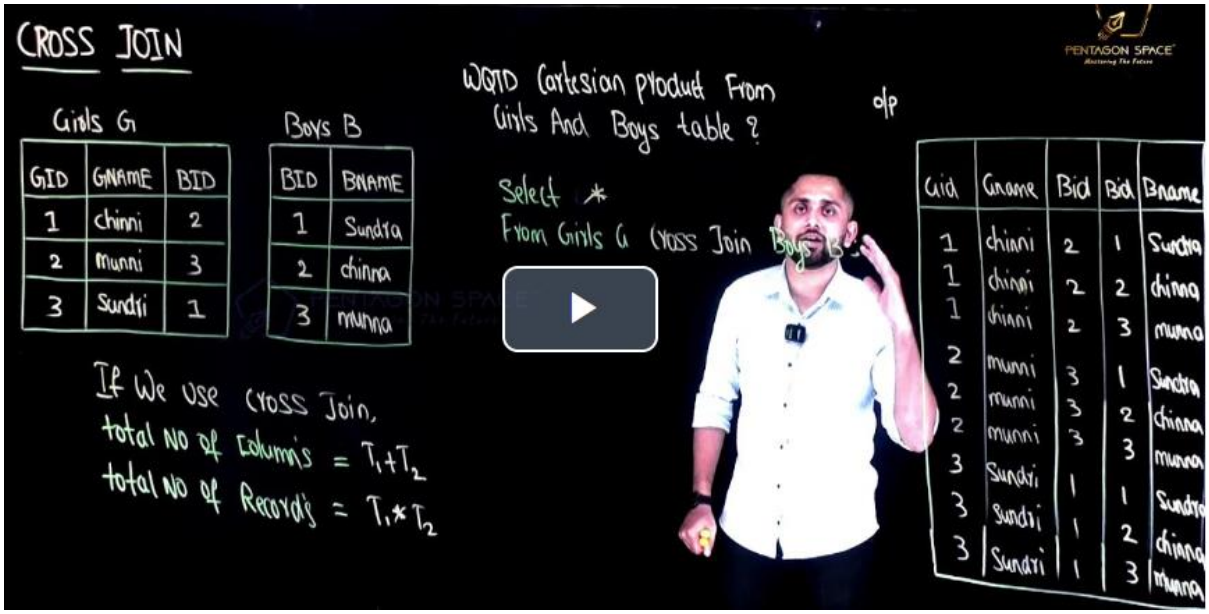
   It is used to merge the records of one table with the records of another table

   Ansi: American National Standard Institute

   Syntax:

   Select column_name / Expression

   From table_name T1 cross join table_name T2;



22/07/25 - Tuesday

Wqtd cartesian product from locations and emps table

select *

-> from locations l cross join emps e;

cross Join Drawback

It will obtain more number of unmatched records compared to matched records

---

2. **Inner Join**

   It is used to retrieve matched records from different tables

   Syntax:

   Select column_name /Expression

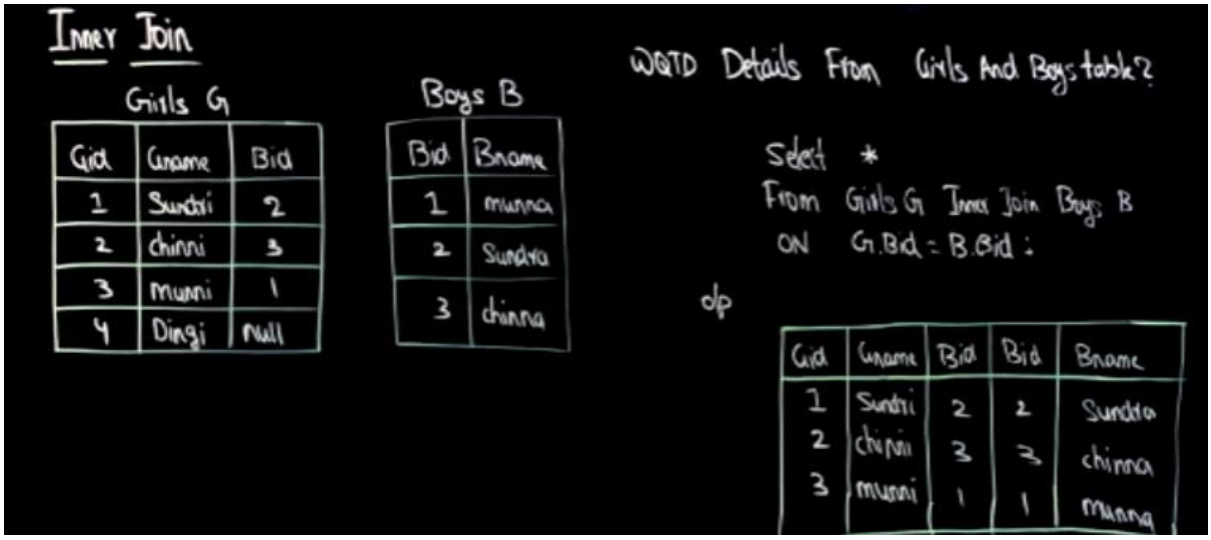
   From table_name t1 inner join table_name t2

   On join_condition

   On: It is used to write the join condition

   Join_condition: It is used to join different tables

   T1.column_name = T2.column_name



Wqtd details from locations and emps table

```
select *
    -> from locations l inner join emps e
    -> on l.lid = e.lid;
```

Wqtd item name along wit restaurant name if item name is masala dosa

```
select m.name as item_name , r.name as restaurant_name
    -> from menu_items m inner join restaurants r
    -> on m.restaurant_id = r.restaurant_id
    -> where m.name = 'masala dosa';
```

Wqtd average rating for each restaurant name

```
select avg(r.rating), rr.name
    -> from reviews r inner join restaurants rr
    -> on r.restaurant_id= rr.restaurant_id
    -> group by rr.name;
```

23/07/25 - Wednesday

Wqtd top 2 restaurants which got highest rating

```
select r.name ,rr.rating
    -> from restaurants r inner join reviews rr
    -> on r.restaurant_id =rr.restaurant_id
```

```
-> order by rr.rating desc
-> limit 2;
```

Wqtd order details along with customer name and delivery person name

```
select o.*,c.name as customer_name,e.fname as delivery_person_name
-> from orders o inner join customers c inner join emps e
-> on o.order_id=c.order_id and o.eid = e.eid;
```

Wqtd customer name who made the highest payment

```
select c.name,sum(p.amount)
-> from customers c inner join orders o inner join payments p
-> on c.order_id=o.order_id and o.order_id = p.order_id
-> where p.status = 'completed'
-> group by c.name
-> order by sum(p.amount) desc
-> limit 1;
```

Wqtd customers name who live in the same city as their delivery person

```
select c.name as customer_name, e.fname as delivery_person_name
-> from customers c inner join locations l inner join emps e
-> on c.lid = l.lid and l.lid =e.lid
-> where e.job = 'delivery' and c.lid = e.lid;
```

Wqtd revenue generated by each state

```
select sum(p.amount) as revenue , l.state as state
-> from payments p inner join customers c inner join orders o inner join locations l
-> on p.order_id =o.order_id and c.order_id = o.order_id
-> where p.status = 'completed'
-> group by l.state;
```

Wqtd monthwise revenue generated in the year 2024

```
select sum(p.amount) as revenue
-> from payments p inner join orders o
-> on p.order_id = o.order_id
```

-> where year(o.order_date)=2024 and p.status= 'completed'

-> group by month(o.order_date);

24/07/25 – Thursday

3. **Outer join**

   i. **Left outer join**

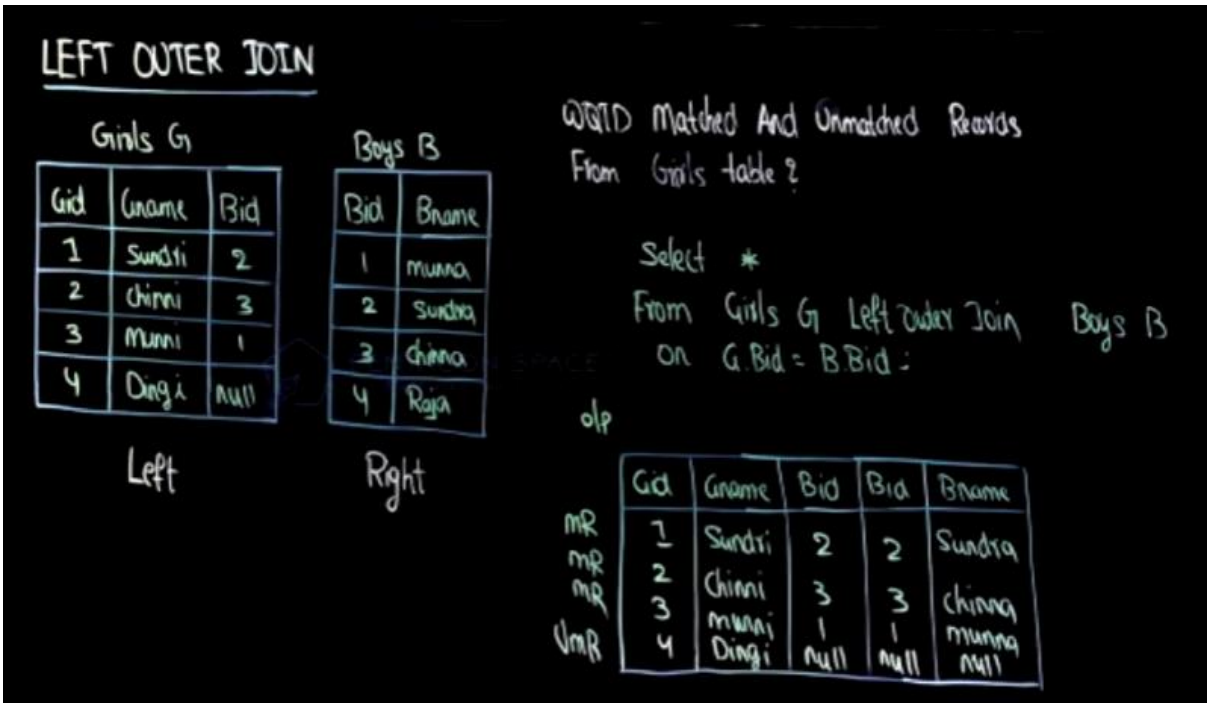
   It is used to obtain matched and unmatched records of left table

   Syntax:

   Select column_name/Expression

   From table_name T1 left outer join table_name T2

   On T1.column_name = T2.column_name ;



Wqtd matched and unmatched records of customers table

select *

-> from customers c left outer join locations l

-> on c.lid = l.lid;

   ii. **Right outer join**

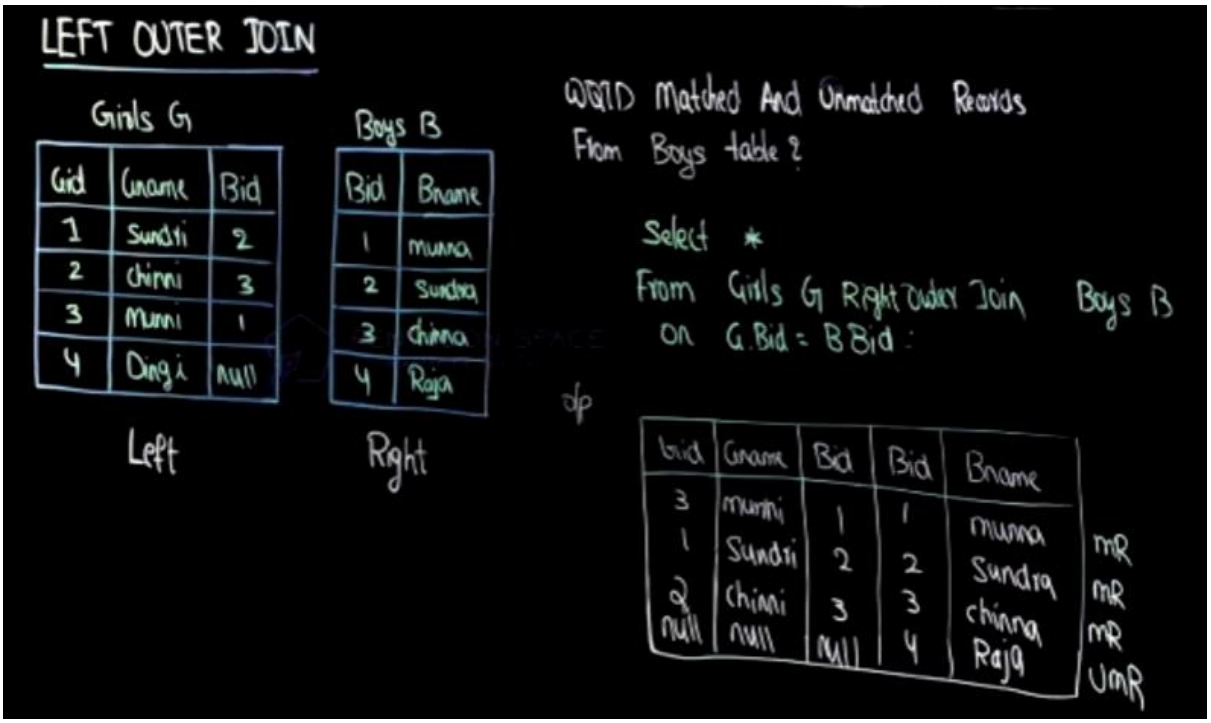
   It is used to obtain matched and unmatched records of right table

   Syntax:

   Select column_name/Expression

   From table_name T1 right outer join table_name T2

On T1.column_name = T2.column_name ;



Customers: left

Locations:right

Wqtd matched and unmatched records of customers table

```
select *
   -> from customers c right outer join locations l
   -> on c.lid = l.lid;
```

Wqtd emps fname who never handled an order

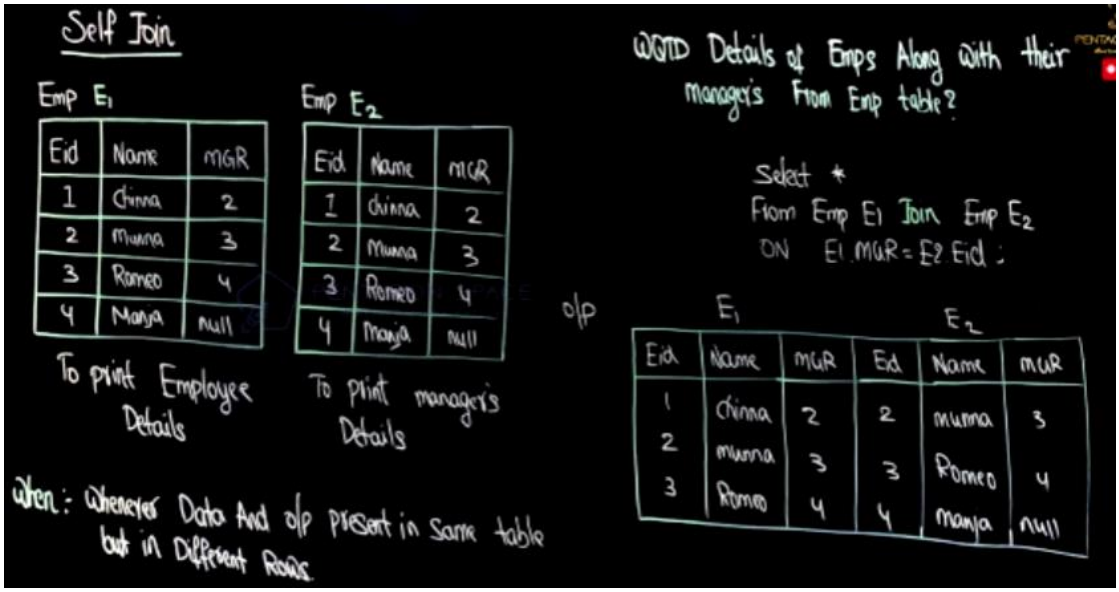
```
select e.fname
   -> from emps e left outer join orders o
   -> on e.eid= o.eid
   -> where o.eid is null;
```

Wqtd restaurant name which don’t have any reviews

```
select r.name
   -> from reviews rr right outer join restaurants r
   -> on r.restaurant_id = rr.restaurant_id
   -> where rr.rating is null;
```

25/07/25 – Friday

4. **Self-Join**
   It is used to obtain matched records from same tables
   Syntax:
   Select column_name / expression
   From table_name t1 join table_name t2
   On join_condition;



Wqtd employee fname along with his manager fname from emps table

E1: For Emps

E2: For Managers

```
select e1.fname as emp_name,e2.fname as manager_name
    -> from emps e1 join emps e2
    -> on e1.mgr = e2.eid;
```

Wqtd emps fname, salary and manger fname, salary if emp is getting salary more than 35000 and manager is getting salary less than 200000

E1: For Emps

E2: For Managers

```
select e1.fname as emp_name, e1.sal as emps, e2.fname as manager_name,e2.sal as
manager_salary
    -> from emps e1 join emps e2
    -> on e1.mgr = e2.eid
    -> where e1.sal>35000 and e2.sal <200000;
```

Wqtd emp fname, dob and manger fname and dob if emp is elder than his manager

```
select e1.fname,e1.dob as emp_dob, e2.fname, e2.dob manager_dob
   -> from emps e1 join emps e2
   -> on e1.mgr = e2.eid
   -> where e1.dob<e2.dob;
```

Wqtd emp fname, job, manager fname , job along with their city name if emp is working as delivery or chef and employee city is either delhi or Jaipur and manger is working as waiter

```
select e1.fname emp_name ,e1.job emp_job,e2.fname manager_name,e2.job
manager_job,l1.city emp_city,l2.city manager_city
   -> from emps e1 join emps e2 inner join locations l1 inner join locations l2
   -> on e1.mgr = e2.eid and e1.lid = l1.lid and e2.lid = l2.lid
   -> where e1.job in ('delivery','chef') and l1.city in ('delhi','jaipur')and e2.job = 'waiter';
```

Wqtd fname of aman's manager's manager

E1:for aman

E2: for aman manager

E3: for aman manager manager

```
select e3.fname
   -> from emps e1 join emps e2 join emps e3
   -> on e1.mgr = e2.eid and e2.mgr=e3.eid
   -> where e1.fname = 'aman';
```

Wqtd details of emps who are getting salary more than kiran

```
select e1.*
   -> from emps e1 join emps e2
   -> on e1.sal>e2.sal
   -> where e2.fname = 'kiran';
```

Wqtd details of emps who are reporting to Priya if Priya is getting salary more than kiran

E1:For Emps

E2: For Priya

E3: For Kiran

```
select e1.*
   -> from emps e1 join emps e2 join emps e3
   -> on e1.mgr =e2.eid and e2.sal > e3.sal
   -> where e2.fname='priya' and e3.fname='kiran';
```

28/07/25 – Monday

5. **Natural Join**

   It is used to obtain matched records from multiple tables

   Syntax:

   Select column_name / Expression

   From table_name t1 natural join table_name t2;

   Example:

   Select * from emps e natural join locations l;

Set operators:

1. Union
2. Union all
3. Intersection

A: {1,2,3,4,5} B: {5,6,7}

A union B : {1,2,3,4,5,6,7}

A union all B : {1,2,3,4,5,5,6,7}

A intersection B : {5}

1. **Union:**
   - It is used to retrieve the data from multiple tables vertically
   - It will avoid the duplicate values from the output
   - **Example:**
     (select fname from emps) union (select fname from emps); o/p : 10 fname

2. **Union all:**
   - It is used to retrieve the data from multiple tables vertically
   - It will involve the duplicate values from the output
   - **Example:**
     (select fname from emps) union all (select fname from emps); o/p: 20 fname

Rules:

- We should use round brackets for queries
- We should use semi colon for the last query
- We should use same number of columns in the select clause

Wqtd job, fname in lower case if the employees are working as waiter else print job, fname in reverse format

```
(select job,lower(fname)
    -> from emps
    -> where job = 'waiter') union
    -> (select job,reverse(fname)
    -> from emps
    -> where job != 'waiter');
```

Wqtd matched and unmatched records from both customers and locations table

Customers: left

Locations: right

```
(select *
    -> from customers c left outer join locations l
    -> on c.lid=l.lid )
    -> union
    -> (select *
    -> from customers c right outer join locations l
    -> on c.lid=l.lid );
```

Wqtd details of 3rd and 6th record from emps table

```
(select *
    -> from emps
    -> limit 1 offset 2 )
    -> union
    -> (select *
    -> from emps
    -> limit 1 offset 5 );
```

Wqtd details of 4th , 5th ,7th and 10th record from emps table

```
(select *
   -> from emps
   -> limit 2 offset 3 )
   -> union
   -> (select *
   -> from emps
   -> limit 1 offset 6 )
   -> union
   -> (select *
   -> from emps
   -> limit 1 offset 9);
```

29/07/25 – Tuesday

Note: By default auto-commit is enable in my sql

To disable autocommit in mysql

set autocommit = 0;

Basically, all ddl commands are auto commit commands

TCL (Transaction control Language)

1. Commit
2. Roll Back
3. Save point

1. **Commit :**

   It is used to save all the transactions (dml operations) permanently inside database

   Syntax:

   commit;

2. **Rollback:**

   It is used to roll out the operations upto previously used commit statement.

   Syntax:

   rollback;

   - **Adv:** we can get the deleted records by using rollback, if commit is not used after delete operation

3. **Savepoint:**

   It is used to mark one position between the transactions

   Data will be saved temporarily, but not permanently inside database;

   Syntax:

   savepoint savepoint_name;

30/7/25 – Wednesday

DCL (Data Control Language)

1. **Grant**
2. **Revoke**

1. **Grant**

   It is used to grant/ provide the permission of the data from one user to another user

   Syntax:

   grant sql_statement on table_name to ‘username’@’hostname’;

   Example:

   grant select on emps to 'pentagon'@'localhost';

   To view all the users present in mysql

   Step1 :

   Use information_schema;

   Step 2:

   Select * from user_attributes

   To view active user present in mysql

   Syntax:

   select user();

   To create user in my sql:

   Syntax:

   Create user ‘username’@’hostname’identified by ‘password’;

```
HOSTNAME:LOCALHOST,%

USERNAME:PENTAGON
HOSTNAME:LOCALHOST
PASSWORD:SQL

CREATE USER 'PENTAGON'@'LOCALHOST'IDENTIFIED BY 'SQL';

TO USE MYSQL ACCOUNT IN COMMAND PROMPT
--------------------------------------
SYNTAX:

mysql -u username -p

GRANT SELECT ON EMPS TO 'PENTAGON'@'LOCALHOST';

GRANT UPDATE ON EMPS TO 'PENTAGON'@'LOCALHOST';

GRANT all ON EMPS TO 'PENTAGON'@'LOCALHOST';

all:TO PASS ALL THE PERMISSIONS AT A TIME
```

Grant all on Zomato.* to ‘pentagon’@’localhost’

Revoke

It is used to get back the permission of the data from another user

Syntax:

revoke sql_statement on table_name from ‘username’@’hostname’;

revoke delete on emps from 'pentagon'@'localhost';

revoke delete on emps from ‘pentagon’@’localhost’;

To drop user from my sql

Drop user ‘username’@’hostname’;

Drop user ’pentagon’@’localhost’;

31/07/25 – Thursday

1. Is it possible to create duplicate table

Yes

Syntax:

create table table_name(select * from table_name);

Example:

Emps1: Emps

create table emps1(select * from emps);

2. Is it possible to create duplicate table without records

Yes….

Syntax:

create table table_name(select * from table_name where false_condition);

Example:

Emps 2: emps

create table emps2(select * from emps where fname='rahul gandhi');

3. Is it possible to add records from one table to another table

Yes…….

Syntax:

insert into table_name(select statement);

Example:

emps2: delivery boys records from emps table

insert into emps2(select * from emps where job='delivery');

insert into emps2(select * from locations where city = 'mumbai');

ERROR 1136 (21S01): Column count doesn't match value count at row 1

Sub-table:

Waiter_data: waiters data from emps table

create table waiter_data(select * from emps where job='waiter');

View:

- It is a virtual table
- It doesn't occupy any memory inside the database
- To overcome te problem of sub table we use view
- Syntax:

  Create view view_name as (select statement);

Waiter: Waiters data from emps table.

create view waiter as(select * from emps where job='waiter');

To drop view

Syntax:

drop view view_name;

Example:

drop view waiter;

1/8/25 - Friday

Corelated subquery

Here both inner query and outer query mutually depends on each other

Working principle of corelated subquery

- First outer query executes partially
- Inner query executes for each record of outer query table
- Outer query executes completely return final output

wqtd employee fname if emps are getting salary more than average salary in their job role

select e1.fname

-> from emps e1

-> where e1.sal>(select avg(sal)

-> from emps e2

-> where e1.job=e2.job);

Wqtd employee fname who is elder than their managers

```
SELECT fname
FROM emps e1
WHERE dob > (
   SELECT dob
   FROM emps e2
   WHERE e2.eid = e1.mgr
);
```

Key Attributes :

The attributes which are eligible to become primary key

Non key attribute:

The attributes which are not eligible to become primary key

Super key attribute :

It is a attribute or combination of attributes used uniquely identified the records

Candidate key:

- It is a smallest subset among key attributes
- In a table we can have multiple candidate key but single primary key
- All the primary keys are candidate key, but all the candidate key are not primary key

Primary key:

A attribute which is used to uniquely identified the records

Foreign key:

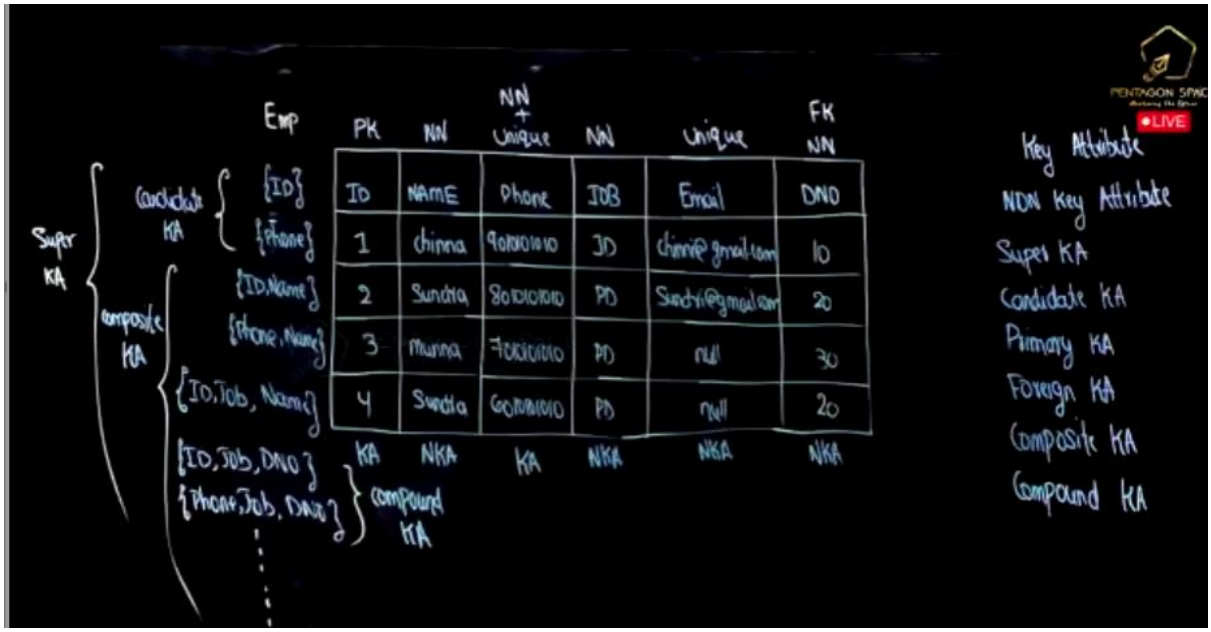
A attribute which is used to estabilish connection between multiple tables

Composite key:

It is a combination of two or more attributes among super key attributes

Compound key:

If the composite key attribute contains atleast one foreign key then we can consider it as a compound key attribute



Ranking Functions/Window Functions

It is used to assign ranks for all records present in table

Syntax:

select ranking_function() over ([partition by column_name] order by column_name asc/desc) from table_name;

over: It is used to pass ranking functions inside select clause.

Partition by : It is used to create the groups, it resets the rank after each group.

Select fname,job,sal,row_number() over(partition by job order by sal desc)

Types of Ranking Functions

1. Row_number()
2. Rank()
3. Dense_Rank()

1. **Row_Number():**
   It is used to assign unique ranks for all the records
   Syntax:
   Select row_number() over[partition by job order by column_name] order by column_name asc/desc)

From Table_name;

Example:

select fname,job,sal,row_number() over(partition by job order by sal desc) 'rank'

-> from emps;

Draw back:

It will assign different ranks for Tied Reords

2. **Rank():**

   It is used to assign ranks for all the records in a table

   Syntax:

   Select row_number() over[partition by by column_name] order by column_name asc/desc)

   From Table_name;

   Example:

   select fname,job,sal, rank() over(order by sal desc) 'rank'

   -> from emps;

   Draw Back:

   It will assign same rank for tied records but it skips next ranking numer

3. **Dense_rank():**

   It will assign ranks for all the records in a table

   It will assign same ranks for tied records also it remains next ranking numbers in sequential order

   Syntax:

   Select dense_rank() over[partition by by column_name] order by column_name asc/desc)

   From Table_name;

   Example:

   select fname,job,sal**,**dense_rank() over(order by sal desc) 'rank'

   -> from emps;

4/08/25-Monday

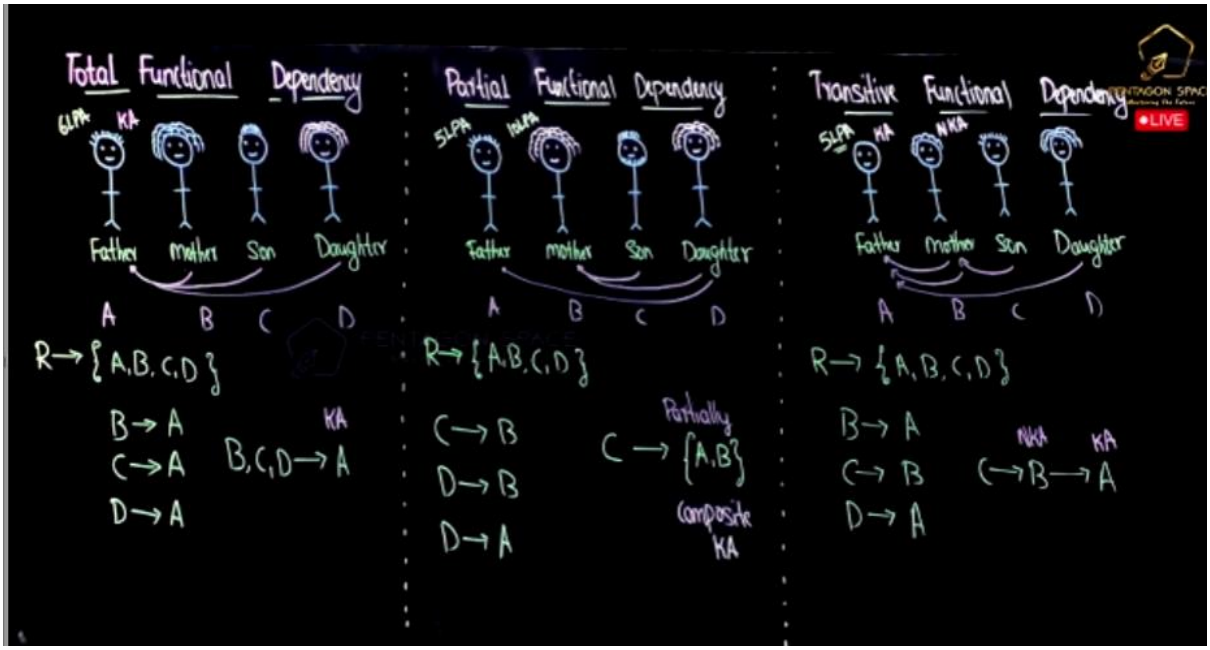
Dependency:

If one attribute depends on another attribute then the process will be known as dependency

Types of Dependency:

1. Total Functional Dependency
2. Partial Functional Dependency
3. Transitive Functional Dependency

1. **Total Functional Dependency**
   In a relation, if all the attributes depends of key attribute then the process will be known as total functional dependency
2. **Partial Functional Dependency**
   In a relation , if one attributes depends on composite key attribute then the process will be known as partial functional dependency
3. **Transitive Functional Dependency**
   In a relation if one attribute is depends on non key attribute which is directly depends in key attribute then the process will be known as Transitive Functional Dependency.



Redundancy:

The process of repetation of data will be known as redundancy

Anamoly:

The problems occurs due to DML operations is known as anomaly

Types of anomaly

1. Insert Anomaly
2. Delete Anamoly
3. Update Anomaly

Normalization:

The process of splitting larger table into smaller tables to avoid redundancy and anomaly is called normalization

Levels: Normal Form

1 NF:

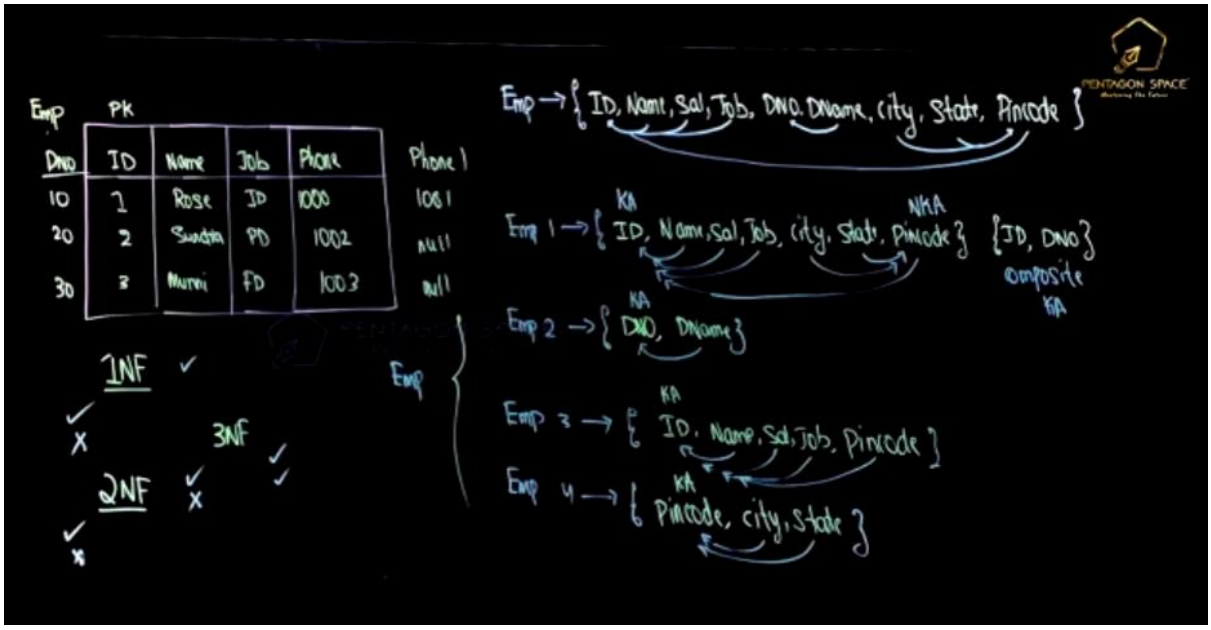
- Table should contain unique records
- A cell in a record must contain single value data

2 NF:

- Table must follow 1NF
- Table should not follow partial functional dependency

3NF:

- Table must follow 2NF
- Table should not follow transitive functional dependency


PENTAGON SPACE
Mastering the Future
Emp
PK
DNO | ID | Name | Job | Phone | Phone1
10 | 1 | Rose | JD | 1000 | 1001
20 | 2 | Swathi | PD | 1002 | null
30 | 3 | Murali | FD | 1003 | null
1NF
2NF
3NF
Emp → { ID, Name, Sal, Job, DNO, Dname, city, State, Pincode }
Emp 1 → { ID, Name, Sal, Job, city, State, Pincode } {ID, DNO} composite KA
KA
NKA
Emp 2 → { DNO, Dname }
Emp 3 → { ID, Name, Sal, Job, Pincode }
Emp 4 → { Pincode, city, State }