NORMALISATION

- Normalisation is a process which is used to organise the data into databases.
- Normalisation are a guidelines which are used to reduce the data redundancy (duplicate data).
- If you are not going to follow normal forms we may get insertion, updation and deletion problems.

-

NORMAL FORMS

1) First Normal Form (1NF)
2) Second Normal Form (2NF)
3) Third Normal Form (3NF)
4) Boyce-codd Normal Form (3.5NF)
5) Fourth Normal Form (4NF)
6) Fifth Normal Form (5NF)

First Normal Form :

For a table to be in 1NF we have to follow below Rules.

(i) There should be no multi-valued attributes. (single atomic values)
(ii) There should be no duplicate columns.
(iii) While inserting the data into a table order does not matter.

Second Normal Form :

For a table to be in 2NF
(i) first it should be in 1NF.
(ii) Every Non-prime attribute must be fully functionally dependent on prime attributes.
(there should be no partial dependency).

prime attribute → which is used to identify unique data
non-prime attribute → which are dependent prime attributes.

Candidate key

- Candidate key means single or minimal superkey.
- While selecting candidate key from superkey, the superkey should not contain null values, duplicate values.
- In one table there can be 'n' number of candidate keys.

- (sid, sname)
- (sid, ~~smarks~~)
- (~~sid~~, sname, Smarks)

sid	sname	Smarks
S1	A	40
S2	A	30
S2	B	40
S3	C	10

candidate keys

{ Sid, Sname, Smarks } => superkey Not Candiatekey.

From the pool of candidate key we select Primary keys.

- Normalisation is
- Normalisation
redundancy
- If you have
insertion

1) First Norma
2) Second Norm
3) Third Norm
4) Boyce - cod
5) Fourth No
6) Jifth Norm

First Norm
For a table
(i)
(ii)
(iii)

Second Norm
For a tabl

(there

prim

Integrity Constraints

- Integrity Constraints are set of rules which is used to maintain Quality of information.
- There are 3 types of integrity constrains
  1) key Constraints
  2) Domain Constraints.
  3) Referential Integrity Constraints

Key Constraints: In a table key attributes are used to identify the unique data.
key Attributes should not contain 'null values'

Domain Constraints: Domain is the value which is present in a table attributes.
- Every attribute bound to have a specific values.
  Sname (Varchar)

Referential Integrity Constraints: By using referential integrity constrains we can refer, one table to another table.
(Here we work on the concept of Foreign key)

Superkey

~~It is a~~ key Attributes are also known as superkey.
- A superkey is a single Attribute or set of Attributes which helps us to identify the unique data from the tables.
- A superkey can contain any number of columns.
- key attributes should not contain duplicate values or null values.

SID	SName	Sage	Smarks
1	A	20	40
2	B	21	20
3	A	20	30
4	C	23	10

sid → superkey
sid, sname → superkey
sid, sname, sage → superkey
sid, sname, sage, smarks → superkey

SUB QUERIES

Single Row SubQueries
When we want to fetch data from unknown data

Multiple Row SubQueries

[Ex] Write a query to retrieve all the employee information whose location is same as ADAMS manager location.

```
set. select loc
from emp
where loc =
```

```
select *
from emp
where deptno = (select deptno
                from dept
                where loc = (select loc
                             from dept
                             where deptno = (select deptno
                                             from emp
                                             where empno = (select mgr
                                                            from emp
                                                            where ename
                                                            = ADAMS)))))
```

- Integrity Constraints are
of information.
- There are 3 types of int
1) key C
2) Domain
3) Refere

Key Constraints: In a tab
data.
Key Attributes should n

Domain Constraints: Dom
attributes.
- Every attribute bound
Snam

Referential Integrity Const
we can refer one tab
(Here we work on

~~it is~~ a key Attributes
- A superkey is a su
Identify the unique
- A superkey can con
- key attributes sho
sid → superkey
sid sname → supe
sid, sname, age →
sid, sname, age.

DCL Queries
(Data Control Language)

1) Grant 2) Revoke.

- DCL queries are used to give the permission for a user by the current user
- There are 2 commands which are used for giving permission or restrict the permission.
- Grant :→ use to give/grant the permission.

  Revoke :→ to take back the permission.
- To see all the user in the current database
  select * from all users.
- to see current user (show user)
- Create user username
  Identified by Password ;
- Switch the User → CONN username/password;

Syntax Grant :-
grant privilage on TableName to username;

Syntax Revoke :-
Revoke privilage on TableName to username;

Syntax for left Join

```
Select
from Table1, Table 2
where Table1.common colum (+) = Table 2.common colum
```

right table — left table

From above syntax all the records of table 2 and common records of table 1 will be displayed.

Syntax for right join

```
where table1.common column (+) = Table 2.common column;
```

All the records of table 1 and common records of table 2.

1] Grant

- DCL queries are
- There are 2 co
the permission
- Grant :→ use
Revoke :→ to
- To see all the

- to see comm
- Create use
Iden
- Switch the

Syntax

Syntax

Inner Join

Inner Join :

Equi Join / simple Join : Equi join is also known as simple join

- in this type of joins, only the common records from both the table will be display

Syntax:

```
table1.column , table.colum ..... table2.column1, table2.column2 ...
    from Table1 inner join Table2 on
        table1.common column = table2.common column ;
```

[#1] Write a query to do the inner join

```
select customer.cid, customer.cname, product.pname, product.price
    from customer inner join product on customer.pid = product.pid;
```

CID	CName	Pname	Price
103	ijk	pencil	5
101	abc	pencil	5
102	xyz	marker	50

A Only those customer record is displayed whose product id is visible

Syntax 2:

```
select columnname
    from table1, table2
    where table1.common column = table2.common column;
```

INE INNER JOIN = cartesian join + condition

```
select customer.cid, customer.cname, product.pname, product.price
    from customer, product
    where customer.pid = product.pid;
```

CID	cname	pname	price
103	ijk	pencil	5
101	abc	pencil	5
102	xyz	marker	50

Full join :

Syntax:-

table1.column, table1.column ... table2.column1 ...
from table1 full outer join table2 on
table1.column
table1.commoncolumn = table2.commoncolumn

- In full join, common records of both the table, only records of left table and only record of right table will be present.
- We can say all the records from both the table will be present.

[#] Write a query to do full join of customer and product table.

→ Customer

CID	Cname	PID	total
101	abc	40	5
102	xyz	30	50
103	ijk	20	5
105	pqr		

Product

PID	Pname	Price
10	Pen	5
20	Pencil	25
50	marker	50
40	laptop	50000

Select customer.CID, customer.Cname, Product.Pname, Product.Price from customer full outer join Product on

CID	cname	Pname	Price
101	abc	Pencil	5
102	xyz	marker	50
103	ijk	Pencil	5
105	pqr		
		Pen	25
		laptop	50000

Inner Join :

Equi Join / simple Jo

- in this type of j
display

Syntax:
table1.
from

[#] Write a query to d

select customer.
from customer

CID	Cname
103	ijk
101	abc
102	xyz

Only these com

Syntax 2: select
fr

INNER

Select customer.Cid
from custom
where

CID	C
103	ij
101	ab
102	xy

[#] Write a query to display all the customer details along with product name which they have purchased.

⇒ select ~~customer~~ customer.CID, Customer.Cname, Product.Pname, Customer.Total
from Customer left join Product on
customer.PID = product.pid;

Result ⇒

customer.CID	customerName	Product.pname	Customer.Total
101	abc	pencil	15
102	xyz	pen	15
105	pqr	pen	15
103	ijk		
106	mno		20

2) Right join : whenever we want all the records of right table and common records of left table then we use right join.

Syntax: select Table1.columnName, Table1.col Table1.columnName
from Table1 right join Table2
on Table1.commoncolumn = Table2.commoncolumn ;

[#] Write a query to display all the product information and also display customer details if they are purchasing the product.

⇒ select Product.PID, Product.pname, Product.price, Customer.CID, Customer.Cname
from ~~Product right join~~ Customer right join Product on
customer.PID = product.pid.

Product.PID	Product.pname	Product.Price	Customer.CID	Customer.Cname
1	Pen	15	102	xyz
1	pen	15	103	PQR
2	Pencil	5	101	abc
3	marker	25	null	null
4	laptop	50000	null	nullvalues.

[#] Write a query to display the serial no and name of the student along with trainer name.

⇒ select Student.slno, student.sname
trainer.tname from Student, Trainer;

Result →

sl.no.	s.name	tname
1	abc	Raveesh
2	xyz	Ravesh
1	abc	Manju
2	xyz	Manju.

We can use 'Alias name' of tables also.

from student s, trainer t. → we can give Alias name.

Outer Joins :-

1) Left Join : Whenever we want to fetch all the records of left table and common records of right table then we go for left join.

Syntax: ~~select~~ *

Select Table1.columnName, Table1.col ..., Table2.columnN
from Table1 left join Table2
on Table1.commoncolumn = Table2.commoncolumn;

Customer

CID	cname	PID	total
101	abc	2	5
102	xyz	1	15
103	Pqr	1	15
105	ijk		
106	mno		40

Product

PID	Pname	price
1	Pen	15
2	Pencil	5
3	Marker	25
4	laptop	5000

"JOINS"

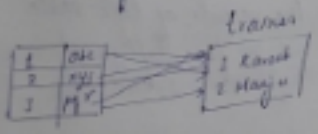
Join → Combining the rows of one or more table is known as Joins.

- Inner Join
  - → Equi Join
  - → Non equi Join
  - → self Join
- Outer Join
  - → Left Join
  - → Right Join
  - → Full join
- Cartition Join / Cross Join
- Anti Join

Cartition/Cross Join : it is used to combine the every rows of table 1 with every rows of table 2.

Syntax: Select *
from Table 1 Cross join Table 2;

syntax: Select *
from Table 1, Table 2;



eg:

Student

S.no	S.Name
1	abc
2	xyz

Trainer

TrainNo	TName
101	Ravesh
102	Manju

select * from Student Cross join Trainer;
OR
Select * from student, Trainer;

O/P ⇒

1	abc	101	Ravesh
1	abc	102	Manju
2	xyz	101	Ravesh
2	xyz	102	Manju

St.no.	Sname
1	abc
2	xyz
1	abc
2	xyz

St.no	trainee
1	Ravesh
1	Ravesh
2	Kishor
2	Kishor

```
insert into Student
values (20, 'xyz', 'D');
```

1 row created

```
Savepoint A;
```

Savepoint created.

```
insert into Student
values (10, 'PQR', 'C');
```

1 row created

```
Savepoint B;
```

Savepoint created.

```
Rollback to B;
```

"TCL Queries"

- TCL stands for Transaction Control languages.
- TCL queries are used to save the transaction or undo the transaction etc.
- We mainly use 3 commands
  1) Commit 2) Roll-back 3] Save-Point

Most used query is 'DML'
DDL ⇒ auto-committed / not auto saving
I have to delete by myself only.

- DDL statement are auto-committed statements but
- DML statements must be saved manually.

1] Commit : it is used to save the transaction.
Syntax : commit;

[*] if we are executing DDL statement after the DML statements then both DDL and DML queries will be auto-committed.

2] Rollback : it is used to undo the transactions.
Syntax : ROLL BACK;
When we do the roll-back operation it will undo the operation upto the last commit operation.

3] Save-Point : They are the intermediate point to which we can do the roll-back operation, instead of roll-back entire transaction.
Syntax : SAVEPOINT 'savepoint name';
eg: Savepoint 'A';
We can do roll-back operation to A savepoint using the syntax.
Syntax : Rollback to 'savepoint name';

* Rollback work Force 'Date' (Not recommended).
Removing the committed record.

Where Clause	Having Clause
(i) It is used on the table rows	It is used on the result after group by.
(ii) We can use a where clause without group by clause.	We can use having clause always with a group by clause.
(iii) In where clause we cannot make use of Aggregate functions.	In having clause we can make use of Aggregate functions.

"Having Clause"

Having clause is used after the 'group by'

```
select *
from table
where condition
group by columnName
having condition;
```

[#] Write a query to display those department which are containing more than 3 employees.

⇒
```
select deptno, count(*)
from emp
group by deptno
having count(*)>=3;
```

[#] Write a query to get the maximum salary of manager, clerk.

⇒
```
Select job, max(sal)
from emp
where job in ('Manager', 'CLERK')
group by job;
```

[##] Write a query to display the minimum salary paid in the department which are located in 'NewYork', 'Chicago'.
Display only those records which contain minimum salary, less than 3000/-
Arrange the result in the descending order of minimum salary.
Use appropriate Alia's name.

⇒
```
Select sal, min(sal) "minimum_salary"
from emp
where loc in ('NEWYORK', 'CHICAGO')
group by desc
having salary >=3000;
```

```
Select deptno, min(sal) "minimum_salary" from emp
where deptno in ( select deptno from dept where loc in
('NEWYORK', 'CHICAGO')) group by deptno
having min(sal) >= 3000
order by min(sal);
```

'GROUP BY CLAUSE'

- Group by clause is used to make the groups based on the distinct values of the column.

eg:
```
select *
from table
where condition
group by columnname;
```

[#] Write a query to count number of employees working in each department.

⇒
```
select deptno, count(*)
from emp
group by Dept No;
```
(Here where condition is optional)

[#] Write a query to fetch maximum salary in each department.

⇒
```
select ~~sal~~ deptno, max(sal)
from emp
~~where sal~~ group by Dept No;
```

[#] Write a query to display minimum salary in each different job profile.
⇒ Arrange the result in an alphabetical order of job profiles.
Use appropriate Alias name.

⇒
```
select job "profile", min(sal) "minimum_salary"
from emp
group by job
order by job;
```

Having clause

[#] Write a query
employees

⇒ sel

[#] Write a query

⇒

[#] Write a
which
Display
Arrange
Use ap

⇒

Select
whe

"Row Num"

- it is a virtual column created by the oracle in the resultant table.
- Row number starts with number 1. and ends at the number of rows of the result.

[#] Write a query to display all the details of employee who are working as 'Manager', clerk. Display top 3 records only.

⇒
```
select * from emp
where job in ('MANAGER', 'CLERK') ~~and~~ AND
rownum <= 3;
```

[#] Write a query to display top 3 highest salaried details.

⇒ select * from ~~emp~~ (select * from emp order by sal desc)
~~where~~ where rownum <= 3

[#] Write a query to fetch top3 highest salary paid by the company.

⇒ ~~select * from~~ ~~select Distinct~~ from (select sal from (select distinct sal
from emp order by sal desc)
where rownum <= 3;

~~Select~~

distinct values ⇒ no duplicates

[##] Write a query to fetch the 2nd highest salaried person details.

⇒ select *
where sal = (select max(sal)
from emp

where sal != (select max(sal) from emp));

select * ~~where~~ from emp where SAL = (select max(sal) from Emp where sal != (select max(sal) from emp));

[##] Write a query to fetch the 2nd minimum salaried person details.

select *
from emp where sal = (select min(sal) from emp
where sal != (select min(sal) from emp));

- it is a virtual column
- Row number starts
result.

[##] Write a query to
clerk. Display
⇒ select
wh

[##] Write a query to
⇒ select * from
~~where~~

#] Write a query to
⇒ select

Select
distinct values

[#] Write a query to fetch all the employees details who are working in NewYork, Chicago.

⇒ ~~Select * from emp where DeptNO = (select dept.~~
(select LOC = 'NewYork' OR LOC = 'Chicago' from dept);

Select * from emp
where deptno in (select deptno from dept where LOC = 'NewYork'
OR LOC = 'Chicago');

By using any operator.

Select * from emp
where deptno = Any (Select DeptNO from Dept where LOC in
('NewYork', 'Chicago'));

[#] Write a query to fetch the employee details who are not working in NewYork and chicago. Don't use Not operator.

⇒ Select * from emp
where deptno != Any (select deptno from dept
where loc in ('NewYork' OR LOC = 'chicago');

[#] Write a query to fetch all the employees details who are getting salary more than 1000/- and working in Boston.

⇒ Select * from emp
where SAL > 1000 AND DEPTNO = (select deptno from
dept where LOC = 'BOSTON');

[#] Write a query to fetch employees who are working in accounting department, sales.

⇒ select * from emp
where Dept = Accounting and (select

where Dname IN 'ACCOUNTING', 'SALES'));

SUB-QUERY

- Query inside the query is known as sub-query.

eg: select *
from tables
where condition = (select ___ from table2);

- the sub-query is known as inner query and inner query will be executed first.

[#] Write a query to fetch all the details of the employee who is getting highest salary. (very important)

Aggregate → select
→ group
→ having

Aggregate cannot be used for 'where' clause.

⇒ select * from emp where sal = (select max(sal) from emp);

[#] Write a query to fetch all the details of the employee who is getting less salary in the company.

⇒ select * from emp where
sal = (select min(sal) from emp);

[#] Write a query to display all the employee detail who are working in sales department.

⇒ select * from emp where
~~department(select dname = sales from dept);~~
dept no = (select dept no from dept where
dname = 'sales');

[#] Write a query to f
Chicago.
⇒ select * f
(s
select * from
where dept no
OR loc

By using any operato
with select * from
where dept

[#] Write a query to f
New York and chica
⇒ select * fro
where d

[#] Write a query to f
name than 1000
⇒ select * fro
where
dept

[#] Write a query
department, sa
⇒ select *
where

Aggregate Functions

(i) sum() (ii) Avg() (iii) count (iv) max() (v) min()

- Aggregate function accepts multiple rows and returns single value.

count function : it is used to count the number of rows in the column.
- it ignores the null values.

[##] Write a query to count number of employees present in the employee table.

→ Select count (ename) we usually go for primary keys because unique and no null values.
from emp;

Select count (*) ⟹ automatically it will go for primary keys.
from emp;

[*] ⇒ when we mention the star, column name will be taken automatically it will always choose the column with the highest key.

Max function : it returns the maximum value from the column.

[##] Write a query to fetch the maximum salary from the emp table.

⇒ select max (salary/sal) from emp;

Min function : it returns the minimum value from the column.

[##] Write a query to fetch the minimum salary from emp table.

⇒ seled min (sal)
from emp;

Date function :-

1) current date → it is used to fetch the current client m/c data
2) sys date → it is used to give the current system data.
3) months_between → it is used to give the no. of months between 2 dates.
4) add_months → it is used to add months to the dates
5) sessiontimezone → it is use to find the timezone of the client machine.

Current date	System date
(i) current date gives the client machine date.	system date gives the system date where ever oracle server is installed.

(ii) If oracle server or database and client are both in same machine then both will give same ans.
both command will give same ans.

current date : select current_date
from dual;

[#] Write a query to display today's date.
⇒ select current_date
from dual;

[#] Write a query to display/fetch the time-zone of client machine:
→ select sessiontimezone
from dual;

[#]

Ag

(i) sum() (ii) Avg()

- Aggregate function accepts m

count function : it is used
- it ignores the null values

[#] Write a query to count
→ select count(
from e

select count
from e

[*] → when we mention
it will always ch

Max function : it return

[#] Write a query to fetch
⇒ select max(

Min function : it return

[#] write a query to fet
⇒ seled m
fr

instr string : it is used to find the position of the sub-string / given string.

instr (source, searching string, position, occurance)

eg. banana
123456
instr ('banana', 'na', 1, 2);

finding the string from.

When occurance is not available it provides '0' value.

[##] Write a query to get the position of string 'na' in the string 'banana'.
→ get the 2nd occurance position.

```
Select instr ('banana', 'na', 1, 2)
  from dual;
```

o/p → 5

[##] Write the query to get all the employees email id domain names.

ename	email
abc	abc@oracle.com
xyz	xyz@oracle.com
nivedita	nivedita@gmail.com
priya	priya@gmail.com
niveditan	niveditan@oracle.com

⇒

```
select instr ('mail
  select substr ('mail', (length-11), 3)
    select instr ('mail' @ , 1, 1)
  from emp;
```

```
Select ename, substr (mail, instr (mail, '@' 1, 1),
       length (mail) - instr (mail, '@', 1, 1)
  from employee;
```

15/5/21

Sub-string : it is used to get the part of the string.

substring(source, position, length)

eg : SQL developer
123456789
5,3

[##] Write a query to get developer from the given string SQL developer.

→ select substring ('SQL Developer', 5, 9)
substr()
from dual table;

Length : length function is used to find the length of the string

[##] Write a query to find the length of the string SQL developer.

→ select length ('SQL developer')
from dual;

[##] Write a query to display the domain name of a employee with the name 'abc' fetch the domain name from his mail id.

→ select substr ('abc', 3,) select substr(mail,
where length('@oracle.com')
from emp;

Select substr (mail , length(mail-11) , 11)
from emp
where ename = 'abc';

↳ length mail
@oracle.com.
(11) position

instr string : it is used to find

instr (source,

eg. banana
123456
find 'ban

When occurance is not

[##] Write a query to get the
→ get the 2nd occurance

Select instr

from dua

[##] Write the query to get

ename
abc
xyz
ravindra
priya
nevedita

→ select instr

select

from

Select ename,

from

[#] Write a query to convert 'Java developer' to 'SQL developer'.

⇒ select replace ('Java developer', 'Java', 'SQL')
from dual;

Lower Case:

change upper case to lower case.

[#] Write a query to display employees working details in following format.

'smith' is working as salesman

display the result all in lowercase only.

⇒ select lower(empname) || is working as ||
lower(job)
from emp;

Concat: it is used in concatination

→ 2 operator allowed.

[#] Write a query to display employees joining details in following format.

' smith joined on 01-Aug-81'

without using || operator.

⇒ select concat(ename, concat('joined on' hiredate))
from emp;

Replace:

(source, oldword, newword)

used to replace the old characters by new characters.

[#] Write a query to convert 'hi good morning' to 'hi good afternoon'.

⇒ select replace('hi good morning', 'morning', 'afternoon')
from dual;

[#] Write a query

⇒ select

[#1] Write a query to find the avg salary of managers find out the result in next higher values.

→ select ceil (avg (sal))
from emp
where job = 'manager';

Character Functions

1) Upper
2) Lower
3) Concat
4) Replace
5) Substring (substr)
6) Instring (Instr)
7) Length

Upper: Used to convert lower case character to uppercase

[#2] Write a query to convert hi to upper case.

⇒ Select upper (hi) → convert to upper.
from dual;

[#3] Write a query to display all emp name in upper case.

→ select upper (ename)
from emp;

[#] Write a query to display the info salary of employees along with the their names
Increase their salary by squaring their salaries.

⇒ select ename, power(sal, 2)
from emp;

Mod :

[#] Write a query to find the remainder after dividing the 15/2

→ select mod(15,2)
from dual;

Floor : (round off the value to lower value)

[#] Write a query to round off the value 2.9.

→ select floor(2.9)
from dual;

[#] Write a query to find the average salary spent by the company on employee display result in rounded form.

⇒ select floor(avg(sal))
from emp;

Ceil : It is used to round off the value to next higher value.

[#] Write a query to find the round figure value of 5.1, round it to next higher value.

→ select ceil(5.1)
from dual;

[#] Write a query to find total amount paid by the company in the form of salary.

→ select sum(sal)
from emp;

select sum(sal) "total amount" from emp (by using alias name)

[#] write a query to find the total expenditure on clerks in the form of salary.

→ select sum(sal) from emp
where job = 'clerk';

Avg:

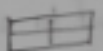
[#] Write a query to find avg salary spent by the company.

→ select avg(sal)
from emp;

Sqrt:

[#] Write a query to find the square root of 9?

→ select square root (9)
from dual;

dual → 
a row/2 columns

Dual - dummy table used when we are performing any operation on literals.

power:

[#] Write a query to find 2^3

→ select power (2, 3)
from dual;

Q/P → 8

SQL - Functions

- It is used to accept one or more parameters and perform some operation and get back some result.

Types of SQL function

1) Single row function
2) Aggregate function
3) Analytical function
4) User defined function.

Single row function : (select/where)

Single row function always returns only one value on row

In single row function we have → 1) Numerical function
2) Character function
3) Date-time function

1) Numerical function :

1) sum
2) avg
3) sqrt
4) power
5) mod
6) floor → from down smaller value
7) ceil → up (biggu value)
8) Random
9) sin
10) cos
11) tan

[#] Write a query to find
salary.
→ select sum
from e
select sum(s

[#] Write a query to
of salary.
→ select
where

Avg :

[#] Write a query to f
→ select
fro

Sqrt :

[#] Write a query
→ select sq
from

Dual - dummy
on literal

power :

[#] Write a query
→ select

Renaming A table

Syntax: Rename Old table to Newtable.

[#] Write a query to change the name of Emp table to employee table.

⇒ Rename Emp to Employee ;

[*] Creating Copy Table

Syntax: Create table newtable as ~~oldtable~~;
select *
from oldtable ;

[#] Write a query to create a table which contains all the managers record refer Emp table.

→ Create table Managers as
select *
from Emp where JOB = 'MANAGER';

[#] Write a query to create a table which contains employe and salary details insert only record who is getting salary more than 2000 /-

→ Create table ~~$~~ 'NewEmployee as
~~Selet~~ select (EmpName, SAL)
from emp
where SAL > 2000 ;

this is the method (Is it possible to insert multiple records at at time)

DML → 1 record. ↓

we can copy it and insert the values.

Set Autotrace ON Explain;

(vii) Index :- By using index we can be able to fetch the records more faster.

- We create the index on a fequently used column.
- Index are created based on 2 types
  - (i) B tree (Binary Tree).
  - (ii) Bit Map
- When we create a index for a particular column the database used index to fetch the record. (not in a traditional way).

syntax: Create index Index-Name on Table_Name (columnName);

-binary search → go to middle.

(viii) Truncate:

syntax: truncate table table_Name

It is used to delete all the record from the table.

[*] Difference between Delete and Truncate. (imp)***

Truncate	Delete
(i) Truncate is DDL command	(i) Delete is DML command
(ii) It will delete is permanent no roll-back operation can be performed.	(ii) Here Deletion is not permanent temporary we can do roll back to retrieve the data.

[##] Write a query to delete all the records permanently from student table without using DML query.

⇒ truncate table Student.

(ix) Renaming A

[##] Write a query

⇒

[*] Creating

[##] Write a que
refer Emp

⇒

[##] Write a qu
travel only

⇒

this is the
DML →

Primary keys	Foreign keys
(i) It is a combination of not null and unique.	In Foreign keys duplicates and null values are allowed.
(ii) In a table only one Primary key is present	In a table more than one Foreign keys can be present.
(iii) In primary key we can insert data which is unique.	In foreign key we can only enter the data which are present in parent table columns.

(V) Check :- It is used to give some condition for the columns like range of value or some static values etc.

```
eg: Create table CareerStudent
    ( Sno int Primary key,
      SName Varchar(50),
      marks decimal
        check (marks >= 35 and marks <= 60));
```

(VI) Default : By using default constraint we can give a default value for a column.

- The default value will be inserted when user is not providing the data for that column.

```
eg: Create table Student
    ( Sno int Primary key,
      SName varchar(50),
      city varchar(50) default 'Bangalore');
```

```
eg: Create table Student
    ( Sno int Primary key,
      Sname varchar(50),
      marks decimal check (marks <= 60)
      city varchar(50) default 'Mumbai');
```

(iv) Foreign key :- Foreign key column data refers the data of another table uniquely.

- Foreign key column can contain duplicate and null values, but it must contain the data of parent column data.
- In a table we can declare multiple columns as foreign key.

```
Create table project (
pjid int primary,
pjname varchar(50) not null,
no of employees int );
```

Table created

```
insert into project
values (10, 'jspiders', 10);

insert into project
values (20, 'qspiders' 20);

insert into project
values (30, 'pyspiders' 30);
```

Value Inserted.

```
Create table employee
Empno int primary,
Ename varchar(30) not null,
Sal decimal,
project id int references Project (Pi id));
```

How to search duplicate foreign keys in table.

```
insert into employee
values (101, 'abc' '5000' 30);
```

- Differences between Primary & Foreign keys.

Primary keys

(i) It is a combination
and unique.

(ii) In a table only
key is present

(iii) In primary key
data which

(v) Check :- It is
range of value

eg: Cre
(

(vi) Default : By
column.
- The default
for that column

eg

eg: Create table
(Slno
Sname
marks
city

Constraints

- Constraints are used on the columns, they are the restrictions for the columns.
- In SQL we have
  (i) not null
  (ii) unique
  (iii) primary key
  (iv) foreign key
  (v) check.
  (vi) Default
  (vii) Index (object)

(i) Not Null :- If any column is declared as not null then data for that column must be provided while inserting a record.

```
Create table spiders (
    sid int not null,
    sname varchar(50) not null,
    marks decimal);
```

not null cannot be left empty.

(ii) Unique :- Any column declared with unique constraints must contain only the unique data, means there is no duplicate values will be allowed in the column.

```
eg: Create table project (
        pjid int not null unique,
        pjname varchar(50) unique,
        size int );
```

(iii) Primary key :- It is a combination of not null and unique.

- In a table one primary key can be present.
- Primary key will help to manupulate individual records in the table.

```
eg: Create table project (
        pj id int primary key,
        pjname varchar(50) unique,
        size int );
```

[#] Write a query to remove the column Email from the student table

=> Alter table Student
drop column Email;

[##] Write a query to change the datatype of ~~Email~~ EName column in the employee table to change it to double.

alter table Emp
modify ename decimal;

Error#: column to be modified must be empty to change data-type.

- Constraints are
columns.
- In SQL we
(i) not
(ii) uni
(iii) pr
(iv) fo

(i) Not Null :-
column m
cr

not null ca

(ii) Unique :-
only the un
in the colu
eg

(iii) Primary ke
- In a table
- Primary ke

insert into student
values (101, 'abc', 25);

(ii) Drop :- It is same as delete operation.

syntax : Drop table TableName ;

- But it will not permanently delete, it will be stored in the recycle bin.
- Delete the table from database.

(iii) Alter :- To do modification in the table we use Alter operation.

Adding column : Alter table Table_Name
add ColumnName datatype constraint ;

Delete/Drop column : Alter table Table_Name
drop column columnName ;

Here column is the keyword.

Rename column : Alter table Table_Name
Rename column oldcolumnName to newcolumnName ;

- column & to → keywords.

Changing Data-type : Alter table Table_Name
modify columName new datatype ;

- Here we can change the data-type only when that column is empty otherwise it will not modify.

[#] Write a query to add a column email to the existing student table.

⇒ Alter table Student
add email varchar(50);

[#] Write a query to change the marks columnname as Scores.

⇒ Alter table Student
Rename ~~Marks~~ column Marks to Scores ;

desc student ;

9/2/21

DDL - Statements

(Data Definition Language)

DDL :- it stands for data definition language

- ddl queries are used to
  - (i) Create
  - (ii) Alter
  - (iii) Drop
  - (iv) Rename
  - (v) truncate
- DDL queries totally deals with the table structure.

(i) Create :- This operation is used to create the table

Syntax: create table Table_Name (
columnName1 Datatype(size) constraint,
ColumnName2 Datatype(size) constraint,
.
.
columnNameN Datatype(size) constraint);

Here constraint is not always mandatory and mentioning size is also not mandatory.

[##] Create a table with the name student and involve the columns s(id), s(name) and s(marks).

⇒ Select * from tab → will show all name of table present.

```
create table student(
    sid int,
    sname varchar(50),
    marks decimal);
```

desc Student;

to get description of a table we need to use the following command desc followed by tableName.

[#1] Write a query to delete employee records whose id starts from 100 to 200

⇒ Delete from Emp
where EmpNo. between 100 and 200;

[#1] Write a query to remove the employee record whose name starts with 'M' and ends with 'R'. and in his name somewhere character 'L' will be appearing.

⇒ Delete from Emp
where EmpName like 'M%L%R';

[##] Write a query to remove all the commission data for those employees who are getting salary less than 1000.

⇒ Update ~~Delete~~ from Emp
Set COM = '0' null
where SAL < 1000;

[##] Write a query to update the table which will have only records of those employee who all hired for between 1st Jan-75 to 1st Jan-95

⇒ Delete from emp
where HIREDATE NOT between '01-JAN-75' and '01-JAN-95';

DML make temporary changes. (Auto-saving is not there)
If I want to save it then commit;

[#] Write a query to increment salary of those employees which are getting salary less than 1000, increment their salary by 5% and add a comm of Rs 100 for them.

⇒ update emp
Set SALARY = (5/100 * SAL/1000), COMM = 100 + COMM
Where SAL < 1000;

Using Insert change in 1 row at a time by using this method.
Update & multiple rows column will change.
Delete

[#] Write a query to decrease the salary of all the salesman and manager by Rs 100.

⇒ update emp
SET SAL = (SAL - 100)
Where JOB in 'SALESMAN' AND 'MANAGER');
OR
where job in ('SALESMAN', 'MANAGER');

[#] Write a query to give a commission of Rs 100 for those who are not getting commission.

⇒ update emp
Set COMM = 100
where COMM is null;

by using 'Is' null value can be selected.

(iii) Delete Operations :- to delete data from the columns we use it

1 Syntax : Delete from table_Name where condition;
[the above syntax will delete all the records which satisfy the conditions]

2 Syntax : Delete from table_Name
If we want to delete every records from the table

[#] Write a query to delete employe

⇒ Delete from emp
where EmpNo

[#] Write a query to remove the
'M' and ends with 'R'. and
will be appearing.

⇒ Delete from Emp
Where EmpName

[#] Write a query to remove
who are getting salary low

⇒ Update Delete from
Set
Where SAL

[#] Write a query to update
Employee who all have

⇒ Delete
Where

DML make temporary
If I want to save it

[#] Write a query to add employee details to the emp table
the details are : id=104, name xyz , job=salesman, Hiredate = 08-Feb-21
sal=8000 deptno=40

⇒ insert into emp (EMPNO, ENAME, JOB, HIREDATE, SAL, DEPTNO)
values (104, 'xyz', 'SALESMAN' '08-FEB-21' 8000, 40);

[##] Write a query to insert the record into the student table with following data
103, Ijk 99.99

→ insert into student
values (103, 'Ijk', 99.99);

[#] Write a query to insert a record into the table the data are
id = 105

→ insert into student (SNO, ~~SNAME~~)
values (105);

(ii) Update Operations :- It is used to update a record into the table

Syntax: update Table
set columnName1 = new value, columnName2 = new value
where condition;

[#] Write a query to change the name of an employee to 'Ijk' who is having emp id = 101

→ Update emp
set ENAME = 'Ijk'
where EMP no = 101;

[#] Write a query to change the employee id to 105 who is having emp id as 104

→ update emp
Set EMP no = 105
where EMP no = 104;

8/2/21

DML Statements

- DML stands for data manupilation language
- It is used for mainly 3 purpose they are
  - (i) insert
  - (ii) update
  - (iii) delete

  these operations on the records of the table.
- DML mainly deals with the table data

(i) Insert Operations :- When we need to insert the values in the table.

1 syntax:-

```
insert into Table_name
    values (column1 value, column2 value ...);
```

2 Syntax:-

```
insert into TableName (column1, column2 ...)
    values (column1 value, column2 value ...);
```

- Syntax 1 is used when we are inserting a record and also we are providing every column data.
- Syntax 2 is used when we are inserting a record and providing only selected column data.

[#] Write a query to insert a column record into the table with following data: id: 101, name: abc, job: CEO, Mgr: 7521, Hire date: '08-Feb-21', sal: 8000, comm: 100, dept No: 40

⇒

```
insert into emp
values (101, 'abc', 'CEO', 7521, '08-FEb-21', 8000, 100, 40);
```

[#] Write a query to ...
the details are :
sal; 8000 ...
⇒ insert into emp
values (...

[#] Write a query to ...
103, ... 9...
→ insert into ...
values ...

[#] Write a query ...
id: 10...
⇒ insert ...
V...

(ii) Update Operations
Syntax: up...
set col...

[#] Write a query ...
having emp id: ...
→ Update em...
set ENAM...
whe...

[#] Write a query to ...
as 104
→ update ...
set E...
whe...

SELECT DISTINCT

Syntax: Select distinct column1, column2 ...
from TableName;

- it is used to fetch unique records from the table.
- if there is multiple columns and we use distinct then a record is said to be duplicate if all the columns data are matching.

[##] Write a query to display the all job profiles available in the company. fetch the details from emp table.

⇒ SELECT DISTINCT JOB
from EMP;

[#] Write a query to display unique records from the EMP table.

⇒ SELECT DISTINCT *
from EMP;

[##] Write a query to display employees details who are getting salary more than 1000 or less than or equal to 3000. Give appropriate alias name and arrange the result in descending order of salary.

⇒ select * from EMP
where SAL > 1000 And SAL <= 3000

ORDER BY SAL DESC;

Select ENAME EMPLOYEE, SAL SALARY from EMP
Where SAL BETWEEN 1000 And 3000
Order by [SAL] Desc;

Order by [SALARY] Desc;

Even if it given by alias name. so we can write. It will give same ans.
→ columnname, columnNo and Alias names.

[##] Write a query to fetch name, salary and commision details of every employees order the result according to 7th column in the actual table.

→ Select ENAME, SAL, COMM from EMP

Order By 7;

column number will take from result so the above is incorrect because we have 3 columns [ENAME, SAL, COMM] so we select *

Select * from EMP
ORDER BY 7;

[##] Write a query to display Ename and commision details accordby to the highest commision to lowest commision order. Display employees details who are not getting commision at the beginning.

⇒ Select ENAME, COMM from EMP
ORDER BY COMM DESC NULLS FIRST;

This keywords make the result to display the null values first

NULLS FIRST
NULLS LAST
— keywords.

Syntax:
select
from
- it is used to fetch
- if there is multi
be duplicate if a

[##] Write a query to
fetch the details
⇒ s

[##] Write a query
⇒

ODER BY CLAUSE

select *

order by columnName1 asc/desc, columnName2 asc/desc ... ;

- Order clause is written at the end of the line.

[##] Write a query to display employee name and his salary details while displaying the result highest paid employees must present at the beginning.

⇒ select EName ~~from~~, SAL from EMP
Order by SAL DESC;

[##] Write a query to display all the employees salary details arrange the result in the descending order of department numbers only fetch name and salary details.

⇒ Select SAL, EName, from EMP
ORDER BY DEPTNO DESC;

Note :- To order the result by any column name, it need not to be present in the select section.
But it will not show in the o/p so we mention in it.

[##] Write a query to fetch highest to lowest paid employees in every department display the result in the order of department number in ascending.

⇒ select * from EMP
ORDER BY ~~SAL DESC~~, DEPTNO ASC, SAL DESC;

first we will sort by department and then salary.
Asc not written still it will take asc. only

[##] Write a query to display all the employee name whose second letter is character 'A'. display result in Alphabetical orders.

⇒ select ENAME from EMP
~~order by ENAME~~
where ENAME LIKE '_A%' order by ENAME;

[#] Write a query to fetch all the employees whose name's first character lies between the alphabet 'M' and alphabet 'Z'

⇒ select * from emp
where EName between 'M' AND 'Z';

[#] Write a query to get all the employees details who joined from January 1st 1981 to July 1st 1981

→ select * from emp
where HIREDate between '01-JAN-81' and '01-JULY-81';
AND

Set Operators :-

1] Union

2] Union all

3] intersect / intersect all

4] Minus.

[#] Write a query to display all the employee name where name starts from alphabet 'A'.

⇒ select * from EMP / select ENAME
where ENAME like 'A%';

[#] Write a query to display the employee name where is having first letter has 'M' and ending with character 'N' and also a letter 'I' will be appearing in his name in the middle position.

⇒ select ENAME from EMP
where ENAME like 'M%I%N';

[#] Write a query to display the job profile name which contains letter 'A' in the second position.

⇒ select JOB from emp
where JOB like '_A%';

[#] Write a query to display the employee name who is having 5 letter in his name.

⇒ select ENAME from Emp
where ENAME like '_____';

5/2/21

Between Operators :- it is used for range of some values
it is used to select the value between the given range

syntax :

```
select *
from table
where columnName between x and y;
```

[#] Write a query to select all the employees who are getting salary in the range of 3000 to 5000

⇒ select * from emp
where sal between 3000 and 5000;

[#] Write a query to display employee details whose job designation is salesman or maybe he is working in department 20.

⇒ select * from EMP
where JOB = 'SALESMAN' OR DeptNO = '20'
from

Since we are using a fields so we cannot use the 'in' operators.

Is Operators :- it is used to compare the null values

[#] Write a query to fetch all the employees all the details who are not getting commissions.

⇒ select * from EMP
where COMM IS NULL;

[#] Write a query to display all the details of the employee who is not having any manager / who is not at all working under any manager.

⇒ select * from EMP
where MGR IS NULL;

Like Operators :- it is used on the columns which contains string data. it is used to give string formats.

- Multiple missing characters are represented by % symbol and single missing characters are represented by _ symbol.

% ⇒ multiple missing characters

_ ⇒ Single missing characters.

[#] Write a query to
alphabet 'A'.

⇒ select
wh

[#] Write a query
letter has 'M'
'I' will be ap

⇒

[#] Write a query
'A' in the

⇒

[#] Write a query
in his name

⇒

5/2/21
Between Operator
it is used
syn

[#] Write a que
the range

⇒

[##] Write a query to display the employee joining information such as joining date and joining department number in the following format.

eg: Smith joined on 17-Dec-88 And working in department no : 20

⇒ select ENAME || ' joined on ' || HIREDATE || ' And Working in ~~dept~~ deptno
Dept No || ' = ' ||
from emp table;

5] Special Operators

in
is
between
Any/some
All
like

in operator :- it is used to select multiple values in a single column

[##] Write a query to display all the employees all the details working in department no 10,20,30.

⇒ select * from EMP
Where Dept NO IN (10, 20, 30);

We can select multiple values in the same column, in different columns we cannot use it

[##] Write a query to display all the details of salesman, clerk and analyst.

⇒ Select * from EMP
where JOB IN ('SALESMAN', 'CLERK', 'ANALYST');

[#] Write a query to display all the details of clerk and Manager who are working in department 10 and 20.

→ select * from EMP
Where JOB IN ('CLERK', 'MANAGER') AND DEPT NO IN ('10', '20');

4/2/21

[##] Write a query to display all the details of manager, clerk and analyst.

⇒ ~~select Job from EName~~
where select *
from ENAME EMPNO.
where Job = 'MANAGER', 'CLERK' 'ANALYST';
OR OR

[##] Write a query to fetch all the employees who gets salary as well as commission.

⇒ Select ENAME,
from EMP table,
~~where = 'SAL' and 'comm';~~
where SAL>0 And COM>0;

4] Character Operators (||) → concatenation

It is used for concatenation purpose, to join the strings or to print details in particular format.

[##] Write a query to display all the employeejob details in the following format.

eg: Smith is working as clerk.

⇒ Select Ename || 'is working as' ||job As "JOB Details"
from emp;

[##] Write a query to display the salary details of every employees in the following format.

eg: Smith is getting salary : 800

select Ename || 'is getting salary :' || SAL As "salary"
from emp;

[##] Write a query
date and
eg: Smi
⇒ select

5] Special Operato
In
is
betw
Any/
ALL
Like

In operator :-

[##] Write a query
department no
⇒
We can select
columns

[##] Write a qu
analyst.
⇒

[##] Write a query
working in
⇒

[##] Write a query to display all employees all the details who are not managers / except Managers.

```
select *
from emp
where JOB != 'MANAGER';
```

3] Logical Operators (AND, OR, NOT)

Here no symbols we have to write word and they are case sensitive.

AND ⇒ it is used when every condition must satisfy

OR ⇒ it is used when anyone of the condition must be satisfy

NOT ⇒ it is used for inverting the conditions.

[#] Write a query to display employees salary details whose salary are in the range of Rs 1000 - Rs 3000/-

```
select ENAME, SAL
from emp
where SAL >= 1000 AND SAL <= 3000;
```

[##] Fetch all the details of employees who are working as salesman, clerk and getting salary more than Rs 1000/-

```
Select ~~ENAME~~ *
from emp
where JOB = 'SALESMAN' OR 'CLERK' AND SAL > 1000
```

[##] Write a query to display job designation detail of smith, Turner & king.

```
select em en
from emp
where ename = 'smith'
```

2] Comparison Operators (>, <, =, !=, <>, ^=)
↓ not operators/not equal

> ⟶ greater than
< ⟶ less than
= ⟶ equal
!=, <>, ^= ⟶ not equal
>= ⟶ greater than or equal
<= ⟶ less than equal

[#] Write a query to fetch all the details of those employees who get salary more than 3000/-

```
select * from emp
where sal > 3000;
```

[We are using * because it is asking all the details of the employees]

[#] Write a query to display department name which is located working in New York city.

→ We must write string data in Quotes
→ the data on the table are case sensitive. As it is we need to write

```
select dname from
  dept
where loc = 'New York';
```

[#] Write a query to fetch all the workers name who are working as salesman.

```
select Ename from ~~Employee~~ emp
where job = 'SALESMAN';
```

[#] Write a query to fetch display all the details of workers where manager having employee id of 7839.

```
select * from emp
where Mgr = 7839;
```

[#] Write a query to di
manager / except Ma

sele
fro
whe

3] Logical Operators
Here no symbols
AND ⇒ it is used
OR ⇒ it is used
NOT ⇒ it is used

[#] Write a query to d
are in the mana

[#] Fetch all the de
clerk and gett

[#] Write a query t

1) Arithematic → (+, -, *, /)

[#] Write a query to display the employe name and his annual salary details from the Emp table.

SQL> select ename, sal * 12 "Annual sab"
from emp;

[##] Write a query to display the employee name and his salary detail, after the increament of 10% for every employees.

select ename, (sal + (sal * 10/100))
from emp;

Alias Name

using Alia name → select emename, (sal + (sal * 10/100)) "hiked sal"
from emp;

[###] Write a query to display employees salary details along with their name and id after a deduction of Rs100 in their salaries.

~~select ename,~~
Select empno, ename, sal-100 "reduced salary"
from emp;

note: If we want to give alias's name which contains space than we must involve alia's name within the double quotes

eg:-

Sl No	S Name	marks
1	abc	99
2	xyz	75
3	pq	85

syntax:-

select sname "student name", marks
from student;

studentname	marks

WHERE CLAUSE

We use where clause to fetch only those records which satisfy the conditions

syntax:
select *
from TableName
where condition;

Operators

1] Arithmatic Operators
2] Comparation Operators
3] Logical Operators
4] Set Operators
5] Special Operators
6] Character Operators

→ These are the operators in the SQL

1 Arithmatic → (+, -, *, /)

[#] Write a query to display the e
from the Emp table.

sql> select ename, sal
from emp;

[##] Write a query to display the
after the increament of 10%

select ename, (sal
from emp;

using Alia name → select ename,
from emp;

[###] Write a query to display em
and id after a deduction

~~select ename,~~
Select empno,
from emp

Alias Name

- It is a temporary name give to columns and tables to display the names properly to display result.
- Alias name does not make any changes in the original table.
- We have to give alias name by using a keyword "as".

[#] Write a query to display all the employees name and their total income detail, use appropriate alias name.

⇒ Select Name as EmployeeName,
Sal as Salary,
COM as COMMISION,
from EMP;

[#] Write a query to display employee name and his designation with his department table using appropriate alias name.

⇒ Select EName as Employee,
Job as designation
dept as department
from EMP;

"as" keyword is not mandatory to give in alias name, we can give space instead of as.

[#] Write a query to display all the records of the department table.

⇒ select * from Dept;

To fetch only the selected fields from the table, we need to use.

Syntax :- select columnName1
columnName2,
column Name3...
from TableName;

[#] Write a query to fetch employeeId, and his Name from the employee table.

⇒ Select EmpNo, EName
from EMP;

[#] Write a query to display the Job designation details of each employee.

⇒ Select Job, EName
from Emp;

[#] Write a query to display employee salary details and his Joining date of each employee

⇒ select EName, sal, HIRE Date
from Emp;

1] SQL :- It is used to retrieve the data from the database

[*] SQL queries are not case sensitive except the table data.

→ All the SQL queries must end with a semicolan(;)

→ All SQL queries execution begins with "from" keywords.

→ If we want to fetch every records from every fields from table then the query will be

```
Select * from Tablename;
```

→ We can fetch all the table name of a account by using following query

→ select * from tab;

tab is a table which contains information about other tables.

Page setup:- to display the result properly in the output screen we need to do page setup.

The following query will be,

Setting the page rows and colounns:

Size :

```
set pages 20;
set lines 200;
```

Relational Database

- In RDBMS, Data is stored in the form of table.
- A single database can have more than one table and they can be related to each other.
- In RDBMS, we can store data more effectively.
- RDBMS provides data redundancy and data security. It is most widely used DBMS.

Tables: Table contains rows and columns

In RDBMS,
Rows are referred as Records
Columns are referred as Fields.

SQL
(Structure Query Language)

It provides some queries to communicate with different types of Database

1) Oracle
2) My SQL
3) MS SQL
4) Derbi
5) IBM-DB2

Types of SQL Queries

1) DQL (Data Query Language)
2) DML (Data Manipulative Language)
3) DDL (Data Definition Language)
4) DCL (Data Control Language)
5) TCL (Transaction Control Language)

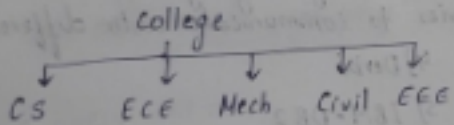
SQL

Data Base :- It is collection of related data.

DBMS :- it stands for 'Data Base Management System'.
- It is used to store, retrieve the data in an Organized manner from the Data Base.
- it is use to manipulate the data in the Data Base.
- DBMS provides a secure connection to access the Data Base by third Party softwares.

Types of DBMS :

1] Hierarchial data Base :- it stores the data in a tree structure.
- In heirarchial DBMS each database are allowed to have a single parent ~~data~~ database.



2] Network Database Model:- In network database model, a database can have multiple parent databases.
- A single database can be accessed from the multiple source through the network
  eg: Gmail. (through the network)

3] Object Oriented Database :- In object-oriented database, data is stored in the form of objects.
- And the collection of objects is called as classes.