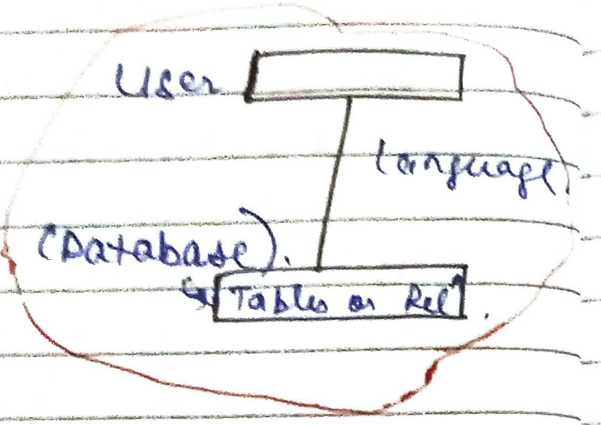
Intro to SQL :- (Structured Query language)

(1970)

→ EF codd give theory of store & fetching of data.

Relational Model:



By the help of Relational algebra & TRC (Tuple Rel. Calculus)

→ IBM converts this concept of database of EF codd into the Structure Query language (SQL).

→ Before,

SEQUEL → Simple English Query language

then,

SQL → Structure Query lang.

→ then, Oracle & Microsoft, etc. comes into this field of databases.

→ Now, No SQL is comes into market.

→ 10% data → Structure
90% data → Unstructure (Spark, Mongodb)

Properties :-

→ General purpose! → Applicability on multiple places. Ex- C/C++, but
Domain Specific! → only use in any particular field. Ex! -
SQL is in only Relational Database. (When we have to store & fetch data from the Relations (Tables), they only we use SQL), and except this, there is no general use of SQL.

1.) SQL is a domain- Specific language.
2.) SQL is a declarative language.

→ declarative lang → what to do.
→ procedural lang → what to do & How to do.
↳ (There is procedure that how to do)

later on,
PL SQL → procedural language.

3.) DDL, DML, DCL, TCL
These are diff. commands in SQL etc. to create, insert, fetch, control data. etc.

4.) keys & constraints. (Ex- P.K, F.K, C.key).
↓
(Primary key (P.K.) constraint,
F.K. "
Not, NULL, default).

aggregate → समान, family, ग्रुप जोड़, संगठन करना।

Exist / Not Exist.

5.) operators (like, between, In, Not in, Conditional).

⇒ We use these operators internally to use Query.

Ex:- IRCTC :- We query on this app, by APIs (Applica' programming interface).

Ex:- Train no', Seat no';
IRCTC based on structured Data.
(Train की Info. Tables के form में ही show होती है।).

6.) clauses (distinct, order by, group by, from having).
Use : majorly use this in query. also

7.) aggregate func's :- (max, average, count, min, sum,)

8.) Joins & Nested Query :-

9.) PL SQL (Triggers, func', Cursor, procedure) same as in C lang.

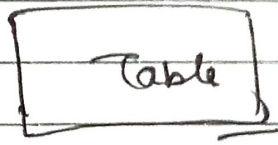
(#) what to do :-
In libraries of oracle or like SQL series, it is already defined that what select do, & what other commands 'do', we don't need to give procedure that how to do.

(27) All Types of SQL Commands :-

* DDL Commands :- (Data Definition language).

→ Deals with Schema (structure | Table).

- → Create
- → Alter (change Table Specifica's).
- → Drop - (Remove Table)
- → Truncate (Remove Data).
- → Rename.



* DML Commands :- (Data manipula" language).

→ If we have to change anything in data, then we use DML. (Manipulation).

- → Select
- → Insert.
- → Update.
- → Delete.

* DCL Commands :- (Data Control language).

→ who has control over the data.
who is authorised on that data.

- → Grant (Give permis")
- → Revoke. (take back that permis").

* TCL commands :- (Transac" control lang.)

→ Mainly used in Transac"s.

Ex: Shopping, Bill payment – online.

- Commit (If Transac^{n} is commit (done), then save data)
- Roll back (If Transac^{n} fails).
- Save point (Ex: we can save after 20-30% of transac^{n}).

Ex:- like in Downloading & Buffering, we have save points. Ex: If downloads fails on 90%, then if it starts from '0' again or from 90%. This is save point.

Note:- To Execute these commands, we can use Oracle, My SQL, Server, etc.

* Constraints :- → We mostly use these Constraints in DDL & DML.

- Primary key (for uniqueness)
- Foreign key (" reference).
- Check
- Unique
- Default
- Not Null (mandatory *).

→ Constraints are rules which we made for store the data.

S3- Create Table in SQL with execution :- →

like SQL

→ In DDL Commands, very first command is Create Table.

→ Now, we use Oracle syntax. (In other like My SQL, SQL server, there is only little diff.).

```
Create table <table-name>          // no space
(                                  ex: abc_xyz.
 Column name  datatype ,
 Column 2 name  datatype ,
 Column 3 name  datatype
);
desc table-name ;                  // describe.
```

Ex:

```
Create table emp
(
id int ,                           → fixed type of datatype
name varchar2 (20),                → variable —//— (can be increased)
Salary number (10)                 ex- (2 bit fixed)
);
desc emp;
```

54. Alter Command (DDL) in SQL :→

Use → (बनी हुई टेबल में अगर कुछ change करना है तो।)

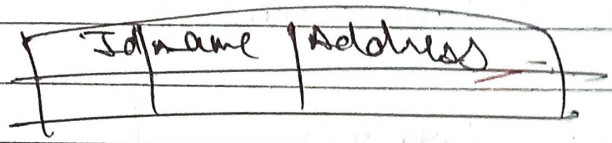
Ex:

Student

int	ID	name	varchar (20)

→ already हुई

→ Add & Remove colmn/s.
→ Modify datatype.
→ Modify datatype length.
→ Add Constraints
→ Remove Constraints
→ Rename col^m n / table.



(4) Alter table student Add address varchar(30);

(, ,);

* multiple colu^m ns.

⇒ Modify datatype:-

Ex:- from int to varchar.

```
* Create table emp
(
id int,
name varchar(10)
);
alter table emp add address varchar(10);
desc emp1;
alter table emp drop column address;
"  "  "  modify id varchar(30);
"  "  "  rename column id to reg_no;
"  "  "  rename to emp1;
alter table emp1 add primary key (reg_no);
```

(55) D/B Alter & Update : →

(*) Alter : only change in structure, (DDL) & not in data.

Ex :

Emp

ID	name	Salary	Email
1	A	10k	
2	B	20k	
3	C	30k	

} → can only change in this struc.

→ Int → varchar

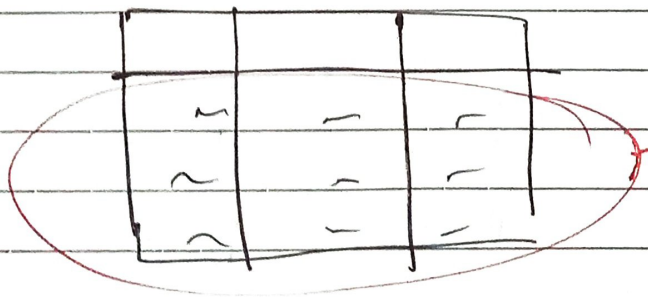
→ name was char (10)
↓
20

→ ID → Eid.

→ Emp → Emp-detail

Colm (name change)
(table name :)

(*) Update : → can only change in data, (DML) & not in structure.



→ can only change in data.

→ Update Emp
Set Salary = Salary * 2 ; → for all students

→ Update Emp
Set Salary = Salary * 2
Where id = 1 ; → for only student of id = 1.

Date :
Page :

56) D|B Delete, Drop & Truncate :-

(1) Delete :-

(DML).

Syntax :-

```
Delete from table name
```

Ex1: `Delete from Student` → All rows deleted.

`Desc Student;`

Student.

ID	name
1	A
2	B
3	C

Note :- But here, also flexibility. We can apply condⁿ for delete some particular rows.

Ex :

```
Delete from Student
Where id = 1;
```

⇒ It is a slower process. - bcz, it also creates log.

> log :- कोई file completely delete होने से पहले, Computer उसे temp. file बनाता है, to restore data. इसे log कहते हैं।

Ex : On delete, our data comes into recycle bin. (It is log file).

→ Rollback;

(possible, but it creates log).

(v) Drop: (DDL) Delete the complete table at once.

```
Drop table student;
Desc student;
```

→ no object found. (all delete).

Rollback; ✗ (not possible).

(*) Truncate: (DDL)

```
Truncate Student;
Desc student;
```

All rows delete at once.

ID	name

→ faster Process.

→ Rollback; ✗ (but no log).

(57) Constraints in SQL: →

→ Constraints means (conditions) restrictⁿ.

Ex: In gmail: (2 account Ids can't be same).

anurag@gmail.com ✗ (not present).

anurag123@gmail.com ✓ (present).

2) Condi^{n} → that new mail-id must be unique.

3) Condi^{n} → Length of password must be 6 digit, Capital letter etc.

Note:- We apply conditions on Attributes (column). and the data which fulfill these condi^{n}, only comes into the columns.

1.) Unique :- → Ex:- Mobile no. (no duplicate can exist in that m.no. col^{m})

2.) Not Null :- *

(We can't skip that col^{m}),

i.e, * - Mandatory. (value देनी ही पड़ेगी)

* 3.) Primary key = Unique + Not Null

Ex:- Reg. no.

4.) Check :- →

age int

:- x (-10)

Check (age > 18) → 10 x
→ 20 ✓

i.e, (2nd particular Domain में से ही User Id चुनी है)

Domain - fixed.

5.) Foreign key :-

6.) Default :-

```
Salary int default 10k
```

If we don't fill anything, they default Salary col^m take value → 10k

⇒ We can use multiple constraint in a col^m also.

58) SQL Queries & Sub-Queries :- (from Beginning to End).

Q:-

Emp.

E_id	E_name	Dept	Salary
1	Ram	HR	10k
2	Amrit	MRKT	20k
3	Rani	HR	30k
4	Nitin	MRKT	40k
5	Varun	IT	50k

(i) Write a SQL Query to display maximum Salary from Emp table.

(ii) Write a SQL Query to Display Employee name, who is taking max^m Salary.

Note :- aggregate func :- for max, count, average.

<> → not equal to, = ≠

(i). Select max (Salary) from Emp;

Output → 50,000

(ii) We here, take help of Nested Query or Sub-Query. (Query inside a Query).

```
Select Ename from Emp where
Salary = (Select max(Salary) from Emp);
```

Outer Query. Inner Query

Output → Varun

⇒ Inner Query execute first 1 time

Ex- 4 EMP

Outer	Inner
10k	= 50k ✗
20k	= 50k ✗
50k	= 50k ✓

(59)

(iii) Write a SQL query to display 2nd highest Salary from Emp table?

(iv) → Write a SQL query to display Employee name who is taking 2nd highest Salary?

(iii).

Logic → Highest Salary - remove &
Rest Salary - find highest.
(It is 2nd highest in table)

Outer

Max ← 10k, 20k, 30k, 40k (50k) → Inner

40k

50k × <

(iii) Query :-

```
Select max (salary) from Emp where
Salary <> (Select max (salary) from Emp);
```

Sol.

Ans

(iv) { = , when we have to compare only with 1 value.
{ in , when we have to compare with multiple values.

Ex :-

2 = 2 ✓

{ 2 = 1,3,2,4,5 (Wrong way)
{ 2 in (1,3,2,4,5) ✓ (True)

Query :- (only add name in (iii))

```
Select E-name from Emp where
Salary = ( Select max (salary) from Emp where
Salary <> (Select max (salary) from Emp));
```

Output will be.

---

60. (v) Write a Query to display all the dept names along with no. of Emps. working in that ?

→

Output :-

HR	2
MRKT.	2
I T.	1

Ex :- In a university, find how many students are there in diff. branches.

AI	101
CSE	100

Note :- Here, we use a special func^n.

[group by] clause.

(forms grp of dept).

HR HR	2
MRKT MRKT	2
IT	1

*** Note :-** Cond^n of group by → जो Att. का नाम हम group by के साथ use कर रहे हैं, सिर्फ वही select ____ के साथ लिख सकते हैं

If,
we have to write other att. in Select other than the att of group by func^n, then we have to use aggregate func^s.

- **Query :-**

Select dept from emp group by dept ;

→ **Output** →

HR
HR
MRKT
MRKT
IT

*** Note :-** Count (dept) or Count (*) → Same.

→ **Query :-** (aggregate func.)

Select dept, Count (*) from emp group by dept;

Output →

HR	2
MRKT	2
IT	1

-ok

61. (ii) Write a Query to display all the dept names where no of Emps are less than 2.

HR 2
MRKT 2
IT 1

Note:- 'Where' clause works on complete Table. but here we group by this table. Hence, group by & where are independent. not help each other.
Hence,
we use 'having'.

Query:-

```
Select dept from Emp group by dept
having Count(*) < 2;
```

Output → IT.

Q ⇒. If they ask of Employee name in this Query, then → (Varun → output)

Query →

```
Select E-name from Emp where dept In
(Select dept from Emp group by dept
having Count(*) < 2);
```

↳ Varun.

In
HR IT
MRKT
IT

62

(viii) Write a Query to display highest Salary department wise and name of emp who is taking that Salary.

Note: A SQL Query always starts from 'from'

Select ___ from table name

Note: Select max (y) from ___ group by x;

Here, It is Right, bcz we use aggregate func 'max' with 'y' i.e. max(y) & if we don't use max, then we can only use x, bcz group by x. i (Emp).

Query :-

```
Select Ename from Emp where Salary
In (Select max (Salary) from Emp
 Group by dept );
```

Output → Ravi, Nitin, Varun ... dt ...

(Steps) →

HR HR	HR - 30,000	30k
MRKT	MRKT - 40,000	40k
MRKT	IT - 50,000	50k
IT		

Outer
10k x
20k x
30k
40k
50k

Inner
30k
40k
50k

* means All attributes.

simple in Query.

63. Use of IN and Not IN :-

I= (1,2,3,4,5) X (Wrong way).

I= 2 false. (but, Right way).

Emp

E_id	E_name	Address
1	Ravi	Chd
2	Varun	Delhi
3	Nitin	Pune
4	Robin	Bangalore
5	Ammy	Chd.

Project

F.K.

E_id	P_id	P_name	Loca"
1	P_1	IoT	Bangalore
5	P_2	Big Data	Delhi
3	P_3	Retail	Mumbai
4	P_4	Android	Hyderabad

Q:- Detail of Emp whose address is either Delhi or Chd or Pune ;

Query:-

```
Select * from Emp where Address = 'Delhi';
```

Output:- | 2 | Varun | Delhi |

⇒ But, we have 3 cities. So,

Query:-

```
Select * from Emp where
        Address In ('Delhi', 'Pune', 'Chd');
```

Output:

1	Ravi	Chd
2	Varun	Delhi
3	Nitin	Pune
5	Ammy	Chd.

Chd ✓
Delhi ✓
Pune ✓
~~Bangalore~~ ✗
Chd. ✓

Nested Query = Query (Query)

Now, NOT IN :- (others not included)

Query :-

```
Select * from Emp Whose
Address Not In ('Delhi', 'Pune', 'Chd');
```

Output :- | 4 | Robin | Bangalore |

Chd ✗
Delhi ✗
Pune ✗
Bangalore ✓
Chd, ✗

64- Use of IN & Not IN in Subquery :-

(Same 2 tables of before).

Query :- find the name of Emps who are working on a project ?

⇒ Suggestion :- First make Dummy Tables.

⇒ Here, we need both 2 Tables.

Both tables have common - EID
So, we compare by taking EID.

⇒ Nested → Bottom up → नीचे से ऊपर

Means पहले अंदर वाली query & then बाहर वाली

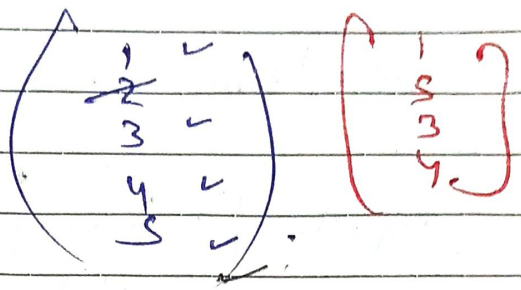
Inner Query in Nested Query - Runs 1 time.

* Query :-

```
Select Ename from Emp where Eid
In ( Select Distinct (Eid) from Project );
```

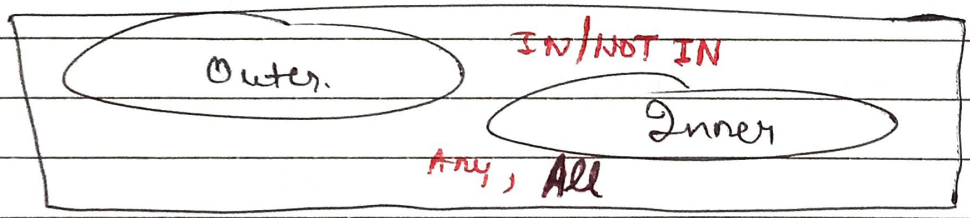
Output :- Ravi
Nitin
Robin
Ammy.

Inner Query.



65. EXIST & NOT EXIST Subqueries :-

↳ We use these in Correlated Nested Query.



⇒ In nested query. — We use IN / NOT IN.

&

In Correlated nested Query - We use EXIST / NOT EXIST.

Query → Find the detail of Emp who is working on at least one Project ?
(Same 2 Tables)

Note :- In nested Query → Inner Query executes first, & then Compare its Output with Outer Query.
In Correlated Nested Query :- the one row of Outer Query is Compared with all rows of Inner Query.
ie, Top to Down Approach.

Null → means value is not available.
Empty

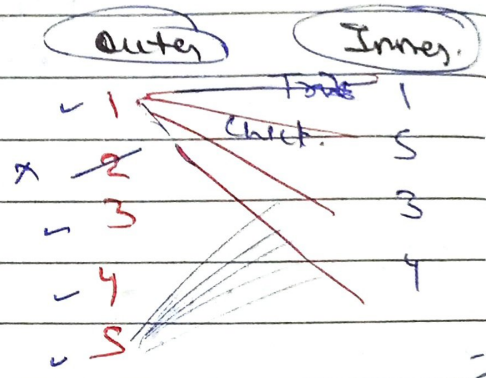
3) Query:-

```
Select * from Emp where Eid
Exists (Select Eid from Project where
        Emp.Eid = project.Eid);
```

Note:- Exist / not Exist gives True or False.

Output:-

1	Rani	Chd
3	Nitin	Pune
4	Robin	Bang.
5	Ammy	Chd.



66) Aggregate Funcs in SQL:- →

Max, Min, Count, Avg, Sum.

↳ (Total no. of values in a Table).

Emp

E_id	E_name	Dept	Salary
1	Ram	HR	10k
2	Amrit	MRKT	20k
3	Rani	HR	30k
4	Nitin	MRKT	30k
5	Varun	IT	50k
6	Sandy	Testing	Null

- Query for Max. Salary:-

→

```
Select Max(Salary) from Emp;
```

Output → 50k

⇒ If 50k repeats 2 times, then it also give output 2 times.

↳ Same for Min.

→ Select Min (Salary) from Emp;

Output: → 10k.

* COUNT:-

Count (*) means no. of rows in the Table.

→ Select Count (*) from Emp;

Output → 6 . 6 rows.

→ Select Count (Salary) from Emp;

Output → 5 (bcz, one value is Null)

* Distinct:-

→ Select Distinct (Count (Salary)) from Emp;

Output → 4 . (bcz 30k is 2 times).

* SUM:-

Slect Sum (Salary) from Emp;

Output → 140 k . (Sum of all salary)

→ Select Distinct (Sum (Salary)) from Emp;

Output → 110 k (30k repeats)

(*) AVG :-

Avg (salary) = Sum (salary) / Count (salary)

= 140K / 5

= 28K

Select Avg (Salary) from Emp;

Output :- 28K.

Select Distinct (Avg (Salary)) from Emp;

Output :- 27,500.

Distinct (Avg (Salary)) = Distinct (Sum (Salary)) / Distinct (Count (Salary))

= 110K / 4

= 27,500

---

(67) Co-related Subquery in SQL :- (Synchronized Query).

- It is a subquery that uses values from Outer Query.
- Top to Down Approach :- (Outer to Inner)
- for One row of Outer Table it compare with every row of inner Table.

→ Returns True / False.

Emp

Eid	name	address
1	A	Delhi
2	B	Pune
3	A	Chd.
4	B	Delhi
5	C	pune
6	D	Mumbai
7	E	Hyd.

Dept

D-id	D-name	Eid
D_1	HR	1
D_2	IT	2
D_3	MRKT	3
D_4	Testing	4

Query:- Find all Employee detail who work in a department.

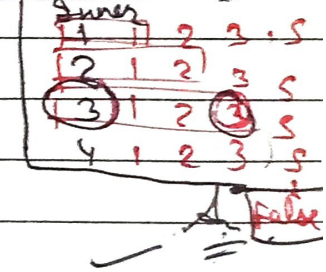
→ True होते ही हमें Employee Detail दे देगा।

→ Query :-

```
Select * from Emp where
    exists ( Select * from dept where
        dept.Eid = Emp.Eid ) ;
```

Output :→

Eid	name	Address
1	A	Delhi
2	B	Pune.
3	A	Chd
4	B	Delhi



68 DB Joins, Nested Subquery & Corelated Subquery :→

→ Nested → Bottom up. (अंदर से बाहर)
→ Corelated → Top down Approach (बाहर से अंदर)
→ Joins → Cross product + Condⁿ

Emp

E_id	name
1	A
2	B
3	C
4	D
5	E

Dept.

Dept no.	name	E_id.
D_1	IT	1
D_2	HR	2
D_3	MRKT	3

Q:→ Detail of all Emp who works in any dept.

(Here, we need both 2 Tables).

→ Nested Subquery →

```
Select * from Emp where E_id in
(Select e_id from dept);
```

Output →

1	A
2	B
3	C

→ Correlated Subquery :→

```
Select * from emp where exists
(select id from dept where
emp.e_id = dept.e_id);
```

E_id is common in both Table.

Output :→

1	A
2	B
3	C

2 rows

1 = 1 → True

2 = 1

3 = 1

2 rows

1 = 5

2 = 5

3 = 5

False

→ If 1st Table → m rows
& 2nd Table → n rows

then, Total Comparison → m × n

→ JOINS :→

Cross product + Condi".

It creates a new Table with (m × n) rows.

→ then,
check Condi" → emp.e_id = dept.e_id

Note:- Joins is faster than Related Subquery bcz it made a table of rows (m × n) at Once. While in Related, we again & again compare one by one row of Outer Table with all Rows of Inner Table. So, it takes time. △ But, Joins take more space. (bcz big Table of (m × n) Rows.).

---

69. Find Nth Highest Salary using SQL :→

Emp.

Id	Salary
1	10k
2	20k
3	20k
4	30k
5	40k
6	50k.

→ Highest Salary :-

```
Select max (Salary) from Emp;
```

Output → 50k

→ 2nd Highest Salary :-
(Nested Query).

```
Select max (Salary) from Emp where
Salary Not In (Select max (Salary) from
                Emp);
```

Output :- 40k

Now

How to recognise Correlated Subquery?
जहाँ पर outer Query का कोई ना कोई value इसने Inner use किया है।

How to find N^th Highest Salary! →

Query :-

Best method → learn it.

```
Select id, Salary from Emp e1 where
N-1 = (Select Count (Distinct Salary) from
        Emp e2 where
        e2.Salary > e1.Salary);
```

↳ Co-related Subqun

Here
we make 2 alias of Emp i.e. e1 & e2
(Dummy)

+ Ex:- on 1st case :-

E_1

ID	Salary
1	10k
2	20k
3	20k
4	30k
5	40k
6	50k.

E_2

ID	Salary	
1	10k	> 10k (false)
2	20k	> 10k count = 1
3	20k	> 10 count = 1 (distinct)
4	30k	> 10 count = 2
5	40k	> 10 c = 3
6	50k.	> 10 c = 4

⇒ Why we make 2 (E_1 & E_2) :-
bcz employee we use 2 times. (E_1, E_2).

↳ If we don't create E_1 & E_2, then
$e_2.Salary > e_1.Salary$
↳ It means, employee salary > employee salary.
↳ no mean. ✗.

↳ E_1 & E_2.

Ex:-
1st
2nd row

10k	> 20k	f
20k	> 20	f
20k	> 20	f
30k	> 20	count = 1
40k	> 20	count = 2
50k.	> 20.	count = 3

} count = 3.

Ex:-) Let we have to find 4th highest, then
N-1 ⇒ 4-1 = 3.
&

```
Select id, Salary
    3 =
```

When we get 3 as count from this Inner query, then, our output comes.

And, we get Count = 3
from the 2nd row bcz,
from 1st Row we got count = 4.
Hence,
Ans is data of 2nd Row

Output:-

ID	Salary
2	20k.

20k is our 4th highest Salary.

50k - 1st
40k - 2nd
30k - 3rd
20k - 4th.

Ex:- for 3rd highest Salary:-
N=3, N-1 = 2

Hence,
for which Row we got Count = 2,
that row will be our output.

10k > 30 F
20k > 30 F
20k > 30 F
30k > 30 F
40k > 30 T , count = 1
50k. > 30. T , count = 2

4th Row is our output

Output: 4, 30k.

30k is our 3rd highest Salary.

3 Imp. Questions on SQL basic concepts :-

Q.(E) You need to display last name of Employees who have 'A' as 2nd character in their names. Which SQL statement display the required result?

a) Select last_name from Emp where last_name like '_A%' ✓

b) — " — last name = '*A%' ✗

c) — " — name like '%A%'

d) — " — like '*A%'

Note:- Questions like, 2nd letter same, Salary be of only 5 nos, like that, based on 'Like' command.

% → any value.
_ → fixed a place for a value.

Ex:- i) %A% Anything AABA ✓ (ये भी सही है)

ii) 2nd letter → '_A%' _ (one pair is fixed)

iii) 4th character must be A → '___A%'

iv) 2nd last letter must be A → '%A_'

(2.) A command to remove relation from SQL database → (Table).

A) Delete table < table name >.
B) Drop table < ——— > 3. Ans
C) Erase table < ——— >
D) Alter table. < ——— >.

→ DDL Commands deal with Schema (Table structure).

Create, drop, Alter → DDL Commands.

→ Alter table → to change anything in table.

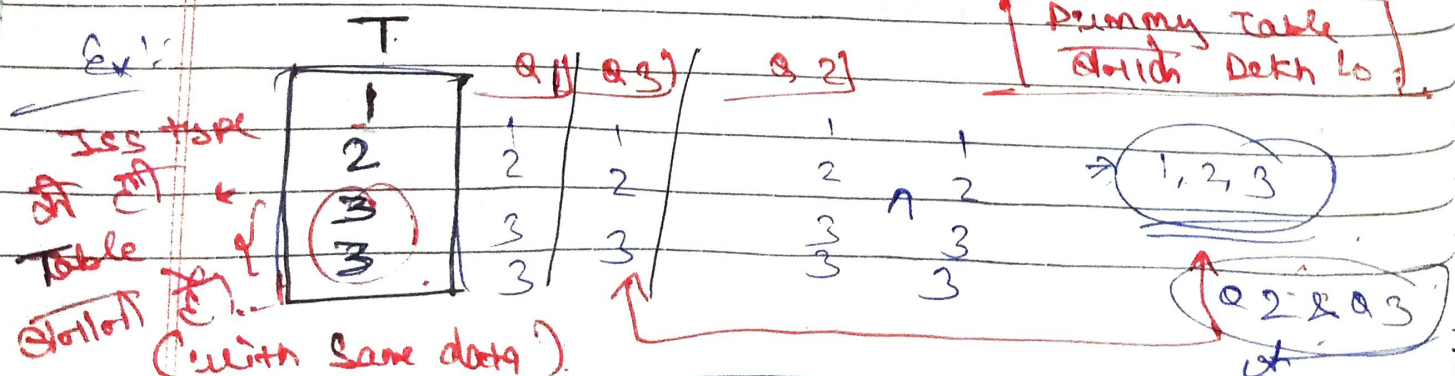
Delete table, Erase table. ≠ (Even not a command).

(3.) In the following, schema R is R(a,b).

Q1: Select * from R
Q2: (Select * from R) Intersect (Select * from R)
Q3: Select distinct * from R

a) Q1, Q2, Q3 produce same result
b) only Q1, Q2 ——— "
c) only Q2, Q3 ——— " ——— 3 Ans.
d) Q1, Q2, Q3 produce diff. result.

Ex:



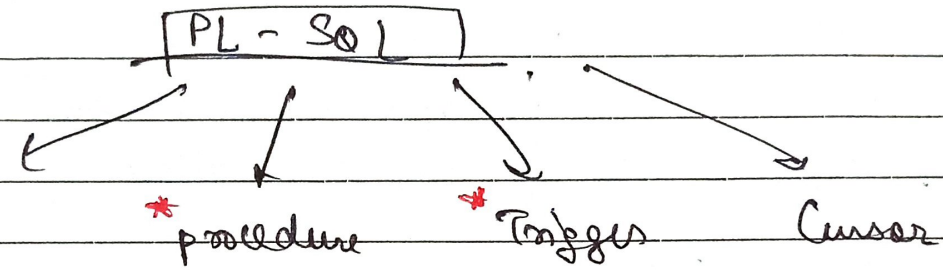
71. PL-SQL

(procedural - SQL).

⇒ SQL → Declarative in nature. (only 'What to Do').

⇒ PL-SQL → procedural (1. What to Do → 2. How to Do.)

(programming flavour वाली चीज़).



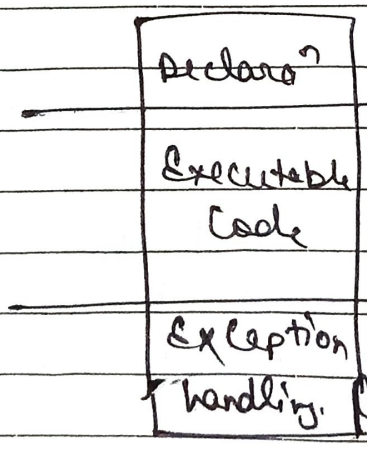
→ In SQL → (Write a query).

→ In PL-SQL → (Write a program, or Write a PL-SQL Code

(Program write करो).

⇒ Its Block Code has 3 parts: →

```
declare
a int
b int
c int
begin
a := 10;
b := 20;
c := a+b;
end;
```



↳ means (If manually, we have to raise any errors). like $\frac{x}{0}$, we define it in Excep^n handling.

in C++ { }