classmate
Date
Page

Welcome To Spiders

APT
G.D
T-1
T-2
HR
Offer

SQL → 5th floor → 8am

SQL (Structured Query language / Sequential Query language)

SQL is used to interact with the Data base

Data base is Backend (Here it is stored in Data centres ocean etc).

SQL

SQL Stands for - Structured Query language
- Sequential Query language

Back end Validation :- The process of Checking weather the front end data is stored in the backend is called as B.E.V

C → Create / Add
R → Read / Retrieve
U → Update / Modify
D → Delete / remove

Operation

- Test Engineers use SQL to do Back-end - Validation
- Interaction with the Database is requireed to do Back - end - Validation.
- The Interaction with Data base are CRUD Operations

CRUD stands for C - Create / Add.
R - Read / Retrieve
U - upload / Modify
D - Delete / Remove.

This is universally called as CRUD Operations.

→ Basic Terminologies

① **Data** :- It is a piece (or) Collection of values use to describe a real world object.

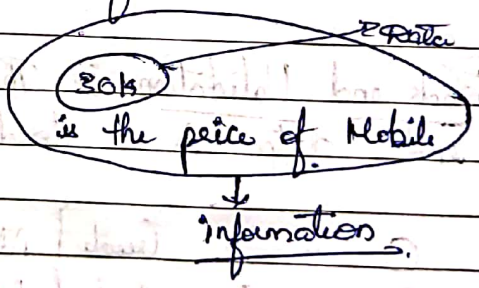
Ex :-

Mobile (Real life object).
- → Price → 30k
- → Brand → Samsung → Data
- → RAM → 6GB
- → Battery → 6000
- → Camera → 48MP.

- Data will become information when it is processed.
- Data will become raw-fact when it is not processed.

Ex :- 30k → data.

Raw-fact.



→ **Note** :- Information Contains Data but Data is not always an information.

② **Data-Base** :- It is a place used to store the data in a systematic and organized manner.

Data-base is a collection of Data.
Data-base is also known as Centralized-place: because all the data is stored in a single place and it

- Can be accessed by all the data-base Users.

- The features of a data-base are :-
  C - Create / Add
  R - Read / retrieve
  U - Update / Modify
  D - Delete / remove.

Examples of a Data-base :- face book, instagram, gmail.
- pendrive, Harddisc / CD

Note :- SQL is Not a programming language, it is a language mainly used to interact with the data-base.
(other programming language is something used to develop Software).

③ ⟹ DBMS :- Data Base Nanagement System.

It is a Software which use query language, by using query language Data base users develop a query, The developed query is used to interact with the data base, to retrive the required data as an output.

- C - Turbo
- C++ - Turbo C++
- Java - eclips / netbeans

DBMS (S/W).
↓
query language. (SQL, AQL etc).
↓
query. (written by Data base users & Test engineer).
↓
data (To interact with data base to retrive data as output).
↓
output.

- DBMS is used to maintain and Manage the Data base.

29,950

Raymond Boyce → father of SQL
E.F Codd → Co-father "

Note :- Data-base & DBMS are interdependent, None can be used alone.

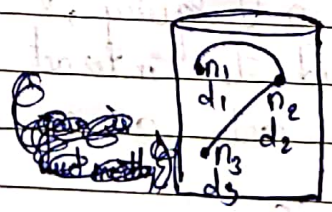
Features of DBMS Software

- C - Create / Add data.
- R - Read / retrieve data.
- U - Update / Modify data
- D - Delete / remove data.
- Providing Security for the data.
- Providing Authentication for Data-base users.

- In DBMS the data is stored in file-format, files may have different formats (or) extensions, Due to that the relationship between the files [Data] can't be established
This is the Main drawback of DBMS.

Note :- Data-Base Models :- 1) Network Model of Database

File ← 2) Hierarchical Model.
Eg Kan Hin 3) Object-oriented Model.
Ram and Arun 4) Relational Model.
↓
Akshaya



node 1 has d_1 & address of d_2 @ n_2, node 2 has d_2 & adress of d_3 @ n_3.

2 Types of DBMS.

1) Networked DBMS. (N-DBMS)
2) Hirerchial (H-DBMS)
3) Object oriented (O-DBMS)
4) Relational (R-DBMS)

EXAMPLE

classmate
Date
Page

④ R-DBMS :- It stands for Relational data base Management System.

- Any DBMS which follows the Relational Model will become of Data base becomes R-DBMS.

Note :- Relational Model was introduced by E.F Codd, where the Data is stored in Table-format.

flow = R-DBMS (S/W) → (only) SQL → query → interact with D.B → retrive o/p

→ R-DBMS is a Software which uses SQL as a language, By using SQL Data base users develop a query, the developed query is used to interact with the data base to retrieve the required data as an output.

- R-DBMS is used to Maintain and Manage the Database.

→ The features of R-DBMS are as same as DBMS. Such as CRUD operation. Authentication & Security.

- In R-DBMS the data is stored in Table format.
- The tables will be having Common-Columns, We apply key-fields key-fields (primary key and foreign key) to the Common Columns and establish the relationship between the Data (Tables)

→ R-DBMS Software Available in the Market.

- Oracle DB → is the Most used in IT Industry (60-70%)
- MY SQL
- MicroSoft SQL Server
- postgre-SQL
- IBM DB-2
- Maria DB

Note :- Oracle Data-base (version-11g) will be used for installation.

Difference b/w

DBMS	R-DBMS
• The Data is stored in file format.	• The Data is stored in Table format.
• Relation-ship between the Data Cannot be established	• Relation-ship between the data Can be established.
• It uses query language	• It uses SQL
• It is used in small organization	• It is used in huge organization
• Normalization Can't be applied	• Normalization Can be applied.

Note :-

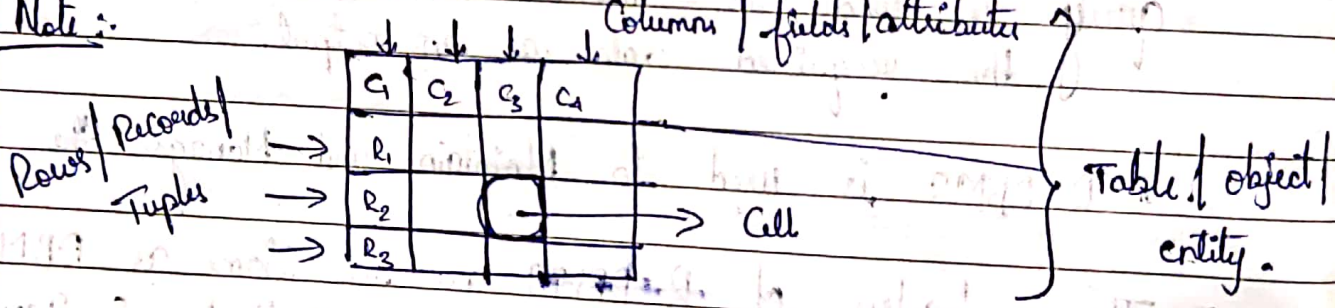


Table :- It is a Combination of Rows & Columns.

Cell :- It is an Inter-section point of Rows & Columns.

Data Integrity :- The process of checking the Correctness and Accuracy of the Data inside the database is known as Data integrity.

- It ensures, only Valid data is stored and invalid data is not stored in the table.
- Data integrity is acheived while storing the data in the R-DBMS Software using the data types and Constraints.

Data-type :- The type of data stored in a particular Column of a table is called as data-type.

- We can assign only 1 Data-type for a single Column
- Data-types are mandatory for all the Columns in a table
- In SQL we use 4- data-types.
  - Char
  - Varchar
  - Number
  - Date

① **Char :-** If a Column is assigned with Character data-type, it can store the following Values.
- alphabets (A-Z Small or upperCase)
- Numbers (0-9)
- Special Characters (#, !, @, #)

- The Default Size for char data type is one (1).

Ex:-

char(3)

C ✓
ABC ✓
124 ✓
a2 ✓
@1A10 ✗

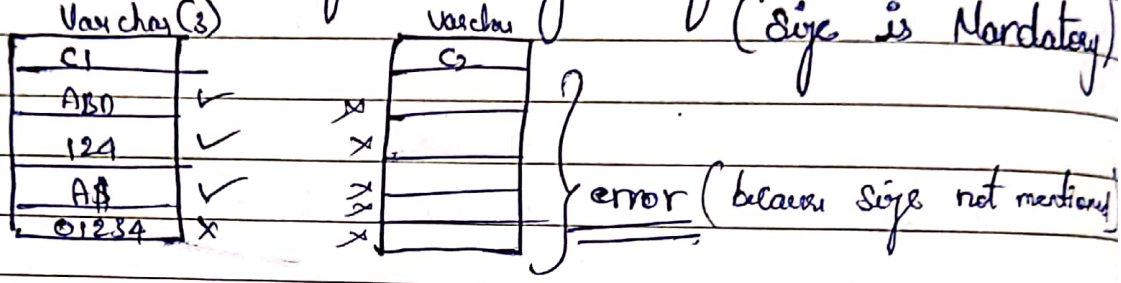
char

C1 ✗
0 ✓
A1 ✗
$ ✓

② Varchar :- If a Column is assigned with Var-char data type, it can store the following values :-

① Alphabets (A-Z upper/lower Case)
② Numbers 0-9
③ Special Characters

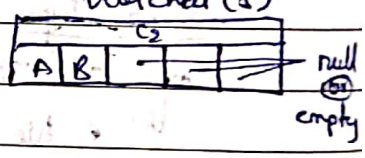
- There is no default size for Var-char. (Size is Mandatory)



Char

- The unallocated memory blocks is filled with space which consume memory
- It follows fixed allocation of memory
- The Default size is one (1)
- We can store up to 2000 characters per cell

Var-char



- The unallocated memory blocks are filled with null which doesn't consume memory
- It follows Variable allocation of memory
- It has no default size
- We can store up to 4000 characters per cell

Note :- Var-char is also known as Var-char2, it has a Maximum size of 10,000 characters per cell.

③ Number :- If a Column is assigned with Number data type, it can store the following values.

① Numbers 0 - 9.

- The default size is 1.

Number (2)	
12	✓
AB	✗
10	✗
-99	✓
0	✓

- Number data type can be used with 2 arguments i.e Number (precision, Scale)
  P S.
- precision :- It represents the maximum number of digits including the decimals.
- Scale :- It represents the maximum Number of decimal values.
- precision - Scale (P-S) = Maximum Number of integers
- precision > Scale (always).

Ex:-

Number (4,2)	P,S
99.99	✓
9.99	✓
9.9	✓
99.9	✓
999.9	✗
9.999	✗
99	✓
9	✓
999	✗

④ Date :- If a Column is assigned with date data-type, It can store only the date values

- The default date format is (DD-MON-YY)
  Ex:- (20-OCT-21) in Oracle DB

Ex:

Dob	
20-OCT-21	✓
Oct-21-20	✗
21-20-OCT	✗
20-October	✗
ABC	✗
'123'	✗

Constraints :- Constraints are the rules of validation applied for columns in a table

- Constraints are optional but highly recommended.
- Multiply Constraints can be applied for a single column.
- The Constraints used are
  1. NOT-& NULL
  2. UNIQUE
  3. CHECK
  4. PRIMARY KEY
  5. FOREGEN KEY

NULL :- It is neither a zero (0) nor a space.

- It won't Consume any memory
- any arithmatic operation performed with null will result in a null.
- In Oracle Null's are not same because they are Compared by their adresses.

	C_1	C_2	C_3
R_1	1	—	2
R_2	3	4	
R_3		6	5

→ Nulls

$R_1C_2 \neq R_2C_3 \neq R_3C_1$

$\therefore$ null $\neq$ null $\neq$ null

Eg:

null + 100 = null.

÷

×

Both are unique because 2 nulls are not same

NOT & NU... it an

UNIQ... it

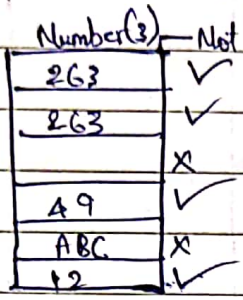
CHE... part

EHRM

EXAM

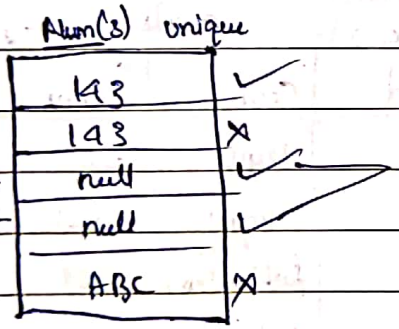
NOT Null Constraint :- If a Column is assigned Not/null Constraint it can't store Null Values (i.e the Column can't have an empty cells).

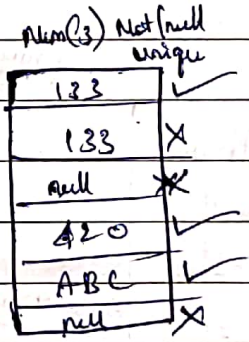
- But the same Column can store duplicate Values.



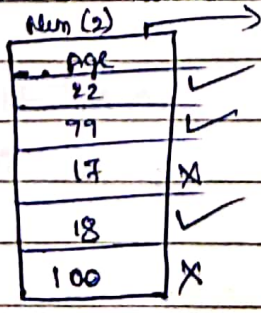
UNIQUE Constraint :- If a Column is assigned with Unique Constraint it can't store duplicate Values.

- But the same Columns can store null Values (i.e 1 or more than 1 nulls).





CHECK Constraint :- It is used to limit the Values for a particular Columns in a table



→ Check (Age >= 18)

- always check comes with check (condition)

- Check Constraint is also called as Domain - Constraint

Domain = It is a set of valid Values.

(Interview Question)

→ Primary Key and Foreign Key

students

Constraints →	NN, Unique	N,N	N,N	N,N, Unique	NN	NN, unique
Data types	Number(2)	Varchar(10)	var char(3)	Varchar(15)	Varchar(10)	number(10)
primary key (PK)	Sid	Sname	Class	mail id	Gender	ph. No.
	01	HHH	LKG	-	-	-
	02	John Cena	UKG	-	-	-
	03	Boogie man	LKG	-	-	-
	04	Lita	UKG	-	-	-
	05	Randy orton	LKG	-	-	-

parent table (or) Master table. — relationship

Sponsors

Primary key (PK) SPid	SPname
01	Red bull
02	King fishes
03	Muthoot finance
04	old monk

parent table / Master table.

relationship

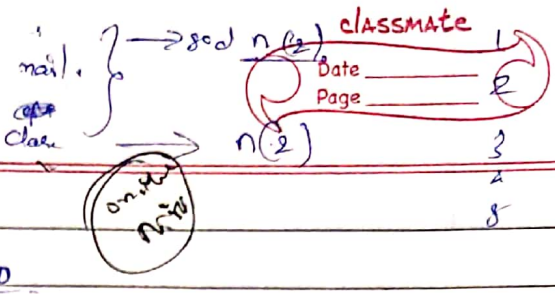
events

Primary key (PK) Eid	Ename	Venue	FK Sid	foreign key (FK) SPid
1	Dance	SIT	04	2
2	Singing	DBIT	04	2
3	fashion show	AIT	03	Null
4	Kabbadi	MSRIT	02	3
5	Snake dance	CIT	Null	4

child table / Slave table.

Primary Key : It is used to identify the records of the table uniquely.

- It is a combination of Not-Null & unique Constraint.
- A Column which are assigned with Not-null and unique Constraint are eligible to become a primary key, and those Columns are called as Candidate keys.



Ex:- Sid, Mail id, & phone No

- The Columns which are eligible become a primary key but not declared as a primary key are called as Alternate keys.

Ex:- Mail-id & phone No.

- A primary key Column can't store Null and duplicate values.
- A Table can contain only one primary key column.
- A Table which contains a primary key Column is called as parent table / Master table.

* Note:- Among the Candidate the Column which is assigned with Number data type with less Size is choosen to be a primary key.

Foreign key :- It is used to establish relationship between 2 (or) more tables.

- If a Column is assigned with foreign-key Constraint, the same Column must be a primary key in another table.
- In other words only the primary key Column of a table is eligible to become foreign key in another table.
- Foreign key Column can store Null, and duplicates but it can't store the values which are not present in the primary key Column.
- A table can have Multiple foreign keys.
- The table which contains a foreign key Column is called as child table / slave table.
- Foreign key is also called as "Referential Integrity Constraint".

SQL (Sub language)

1) DQL - Data Query language / Data Retrieval language (DRL)
2) DDL - Data definition language.
3) DML - Data Manipulation language.
4) TCL - Transaction Control language.
5) DCL - Data Control language.

- Each and every Sub language is having Certain statements

① DQL/DRL - Select (Read (or) retrive)

② DDL - Create, Alter, Rename, Truncate, Drop.

③ DML - Insert, update, delete.

④ TCL - Roll back, Commit, Save point.

⑤ DCL - Geront, Revoke.

→ Data Query language (DQL) - It is used to fetch (or) retrive the data from the data-base.

Select :- It is used to fetch (or) retrive the data from a particular table in the data-base.

- Select statement has 3 possibilites a) projection
  b) Selection
  c) joins.

a) projection :- Fetching (or) Retrieving the data, from a particular table by Selecting all the Columns (or) only the required Columns is Called as projection.

- By default all the rows will be selected.

• We can restrict the Columns but not rows.

$C_1+C_2+C_3$ + $R_1+R_2+R_3$ | C_3 + $R_1+R_2+R_3$ | C_2+C_3 + $R_1+R_2+R_3$

Syntax

② → Select * (specific) / [DISTINCT] Col_name(s) / Expression [ALIAS]

① → from table_name;

order of execution

EX:-

Students

S.id	Sname	branch	Per
1	Yash	MECH	70
2	Darshan	Civil	90
3	Savathri	CS	40
4	Pishapatari	EC	60

Table in a database

→ Write a query to display the names of all students. (WAQTD)

Select Sname
from students;

o/p

Snam
Yash
Darsha
Savat
Disha

Extra

Select Sname, branch
from students;

o/p

Snam	branch
Yash	ME
Dars	CE
Sav	CS
Dip	EC

WAQTD the name & %

Select Sname, per
from students;

o/p

Snam	Per
Yash	70
Dars	90
Sav	40
Dip	60

→ WAQTD id, branch & per of all students

Select sid, branch, per
from Students;

→ WAQTD branch, %, id & names for all students

Select branch, per, sid, Sname
from students;

→ WAQTD the details for all the stude

Select Sid, Sname, branch, per
from Students;

(or)

Select * → represents all the columns in same order present.
from Students;

→ WAQTD the student details along with branch for all students

Select *, branch → This combination is not possible in SQL. ∴ ~~select~~

from Students;

note
Select tablename.*, Column name
can be used. ∴ Select Students.*, branch
from Students;

Software installation process — Mobile SQL plus (Mobile)

→ EMP Table

① → WAQTD the names for all employees.
② → —u— designation, Employee Id/No, & Salary for all employees
③ → —u— Manager No, date of joining, commission —u—
④ → —u— Employee names & dept No for —u—
⑤ → —u— Display the employee details for —u—
⑥ → —u— —u— employee details along with Salaray —u—

① Select ENAME
from emp;

② Select JOB, EMPNO, SAL
from emp;

③ Select MGR, HIREDATE, COMM
from emp;

④ Select Ename, DEPTNO
from emp;

⑤ Select *
from emp;

⑥ Select emp.*, SAL
from emp;

→ SQL is a case insensitive language.

→ DEPT table

① WAQTD Dept name of all employee.
② —11— No, loc —11—
③ —11— dept details —11—

① Select DNAME
from dept;

② Select DEPT NO, loc
from dept;

③ Select *
from dept;

→ SAL grade Table

① WAQTD the lowest & highest salary for all employees
② —11— the grade for all employee

① Select LOSAL, HISAL
from Salgrade;

② Select Grade
from Salgrade;

Note:- * To display the list of tables in a data-base, we use

Select *
from tab;

* Set Command :- It is used to set the line size & page size.

Ex:- Set lines 100 pages 100
Select *
from emp;

* **Describe Command** :- It is used to describe the table structure

Ex:- describe emp; (i.e it describes Constraints & Datatypes present for apartticular columns).

* **Connect Command** :- It is used to switch between the database users. i.e (Scott & HR).

Ex:- Connect

→ **Distinct clause** :- It is used to remove the duplicates in the output.

students

SName	branch	%
A	CS	60
B	EC	70
C	CS	60
D	CS	75
E	Mech	100
F	EC	70

① WAQTD the unique branch

```
Select distinct branch
from students;
```

o/p

branch
CS
EC
mech

② WAQTD the unique Combination of branch & Percentage.

```
Select distinct branch, per
from students;
```

o/p.

branch	per
CS	60
EC	70
CS	75
Mech	100

- It should be used before a Column name.
- It removes the duplicates only in the output, but not in the table.
- Distinct Clause is not mandatory
- If a single Column is used in distinct, unique values will be provided only for that Column.
- If multiple Columns are used in distinct, unique Combination of values will be provided.

① WAQTO the unique department number values from emp table.
② — " — Salary values without duplicates — " — .
③ — " — unique combination of job & department No — " — .
④ — " — " — of Salary & Manager No — " —

Ans :-

① Select DISTINCT dept NO.
from emp;

② Select distinct SAL
from emp;

③ Select Distinct job, dept No.
from emp;

④ Select distinct MGR, SAL
from emp;

→ Expression clause :- It is a combination of operators and operands.

Ex :- a (+) b — operands
operator

• Operands :- They are the inputs used in an expression.

There are Two - Types of Operands.

1) Column Type Operand.
2) Litural Type Operand.

• Column - Type operand :- If a Column name is used as an input in the expression, it is called Column - type operand.

Ex :- sal + 100
↓
Column type.

• Litural - Type operand :- If a direct value is used as an input in an expression it called litural - Type operand.

0-025 X 100

classmate
Date
Page

Litral type operand is classified in to 3-Types

a) Number - litral ⟶ Ex:- Sal + (100)
↳ Number litral.

b) Date - litral ⟶ Ex:- hire date + '27-OCT-21'

c) Character litral ⟶ Ex:- Ename + 'ABC'.

Note:- Date - literals and character literals should be used with in Single Quotes ' '.

① WAQTD the annual salary for all the employee
② —u— half-tum Commision & Annual commision for all the employe
③ —u— the Salary with bonus of 500 & with penalty of 200.
④ —u— 25% hike of the Salary for all employee.
⑤ —u— 75% deduction & 50% hike of the Salary.

① Select sal x 12
from emp;

② Select COMM x 6, COMM x 12
from emp;

③ Select sal + 500, sal - 200
from emp;

④ Select (0.25 x sal) + sal
from emp;

⑤ Select sal - (0.75 x sal), (0.5 x sal) + sal.
from emp;

⑥ WAQTD 25% hike in annual salary.
Select sal x 12, sal x 12 x 0.25
from emp;

→ Alias Clause :- It is an alternate name given for expression, Column name & functions.

- It is not mandatory, but used to get user Convenient output.
- If Alias Contains any Special Character it should be used with-in Double Quotes " "
- We Cannot use Keyword of SQL as an Alias

Syntax :- Select expression Alias /
function Alias /
Col-name Alias /
from Table-name;

→ Space (or) as

select expression as Alias
function as Alias
Col-name as Alias.
from table name;

Ex:- ① Select ename as empname
from emp;

② Select (Sal - Sal * 0.75) as "75% deduc"
from emp;

③ Select Sal - Sal * 0.75 as "75% deduc", Sal + 0.5*Sal as "50% hike"
from emp;

Assignment :-

→ No space in alias key word

① Select Sal+500 as salhike, Sal-200 as Saldedu
from emp;

② Select Sal+(0.25*Sal) as Sal hike.
from emp;

③ Select Sal*12 as annualsal,
Sal*12+(0.25*sal) as
Sal hike
from emp;

④ Select Sal-(0.7*Sal) as Salded, (0.5*sal+Sal) as Sal hike
from emp;

b) Selection :- Fetching (or) retrieving the data from a particular table by selecting all the Columns (or) required Columns along with the required rows is called as Selection.

- Here both Columns & rows can be restricted.

Syntax :- ③ Select * / [DISTINCT] Column name / Expression [ALIAS]
① from table_name
② where < filter Condition > ;

② where → order of Execution

< filter Condition > → LHS (=) RHS

= any operator

→ where clause :- It is used to filter the records of the table.

- It works based on the conditions specified
- It works in the format of Row-By-Row, (i.e. the Condition will be checked for each & every Row).
- If the Condition is true where clause will select the (entire) record.
- If the Condition is false where clause will reject the (entire) record.
- The format for the Condition is LHS (operator) RHS

Note :- In SQL Both literal and character literal are Case Sensitive.

EX :- WAQTD name of the player playing for RCB.

"25% hike"

IPL.

Pno	Pname	team	Runs
17	ABD	RCB	133
7	Dhoni	CSK	90
18	Smriti Mandana	CSK	75
93	Bumrah	MI	10
1	KL Rahul	RCB	119

```
(3) select Pname
(1) from IPL
(2) where team = 'RCB';
```

o/p of where clause is.

Pno	Pname	team	Runs
17	ABD	RCB	133
1	KL Rahul	RCB	119

→ i/p to select

final o/p.

Pname
ABD
KL Rahul

(2) WAQTD the team & Runs scored by the player Smriti Mandana.

```
select team, Runs
from Ipl
where Pname = 'Smriti Mandana';
```

(3) WAQTD the player details who has scored More than 90 runs.

```
select *
from Ipl
where Runs > 90;
```

→ Number literal

(5) WAQTD the dept No for SMITH
(4) —" — Employee ALLEN'S SALARY.
(5) —" — Employee names & Date of joining working in Dept No 30
(6) —" — Employee details for Salesman.
(7) —" — SAL & COMM for the Employees reporting to 7698.
(8) —" — Employee No for the Employees who joined on 02-APR-81
(9) —" — Employee names earning atleast 2000 as Salary.
(10) —" — designation & Manager No for the Employee id 7654.

(11) ——" —— ~~Employee details~~ dept No for the employees who joined after the year 1981.

Answer

(5). Select DEPT NO
from emp
where. ENAME = 'SMITH';

(4) Select SAL.
from emp.
where. ENAME = 'ALLEN';

(3). Select Ename, Hire date
from emp.
where. Dept No = 30;

(6). Select *
from emp
where JOB = 'SALESMAN';

(7). Select SAL, COMM
from emp.
where MGR = 7698;

(8) Select EMPNO, Ename
from emp.
where. HIREdate = '02-APR-81';

(9). Select Ename.
from emp.
where SAL >= 2000;

(10). Select Job, MGR
from emp
where EMP NO = 7654;

(11) Select dept No.
from emp
where Hiredate > '31-DEC-81';

(12) WAQTD the employee details for all employee except smith

Select *
from emp
where ename != 'SMITH';

(13) WAQTD the employee details using where Condition
Select *
from emp
where 1=1;

→ Order by clause:- It is used to arrange (or) sort the output in Ascending (or) descending order.

- Order by clause affects only the output but not the table data.
- It always executes at the last
- It should be the last statement in a query
- By default it sorts the data in ascending order.

Syntax:

③ Select * / col-name.
① from table-name.
② [where < filter condition >]
④ Order by ref-col [ASC] | DESC ;

order of execution

① MAQ I D the Salary Values in Ascending order.
② —"— the Commision Value in Desc —"—.
③ —"— the employee names & Salary only for clerks & sort the o/p with respect to Salary in descending order.
④ —"— the employee name in Alphabatecial order.

Ans

① Select SAL
From emp.
order by SAL ;

② Select COMM
from emp
order by COMM DESC;

③ Select Ename, SAL
from emp.
where JOB = 'CLERK'
order by SAL DESC;

④ Select Ename
from emp.
order by Ename;

⑤ WAQTD the annual Salary in Asc. order for all the employee only if annual Salary is greater than 15000.

```
Select SAL * 12
from emp
where SAL*12 > 15000
order by SAL*12 ASC;
```

(or)

```
Select SAL*12 AS ANNSAL
from emp
where SAL*12 > 15000
order by ANNSAL;
```

⑥ Note :- Alias name can't be used in where clause, but can be used in Order by clause.

Operators :- It is used to perform a particular operation.

Types of Operators :-

1) Arithmatic → +, -, *, /, %.

2) logical → AND, OR, NOT.

3) Relational / Comparison → =, !=, >, <, <=, >= (single valued operators)

4) Set operators → union, intersect, minus, union all.

5) Concatenation → (||)

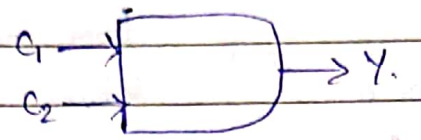
6) Subquery Operators → ANY, ALL

7) Special Operators → IN, NOT IN, BETWEEN, LIKE, IS.
NOT IN BETWEEN, NOT LIKE, IS NOT.

1) Arithmatic :- Addition, substraction, multiplication, division & %.

2) Logical Operator :- It is used to specify multiple conditions in where clause.

① AND

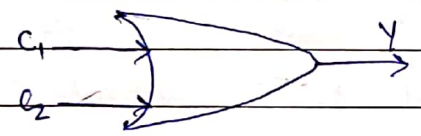


C_1	C_2	$Y = C_1 \& C_2$
F	F	F
F	T	F
T	F	F
T	T	T

Syntax

```
Select * / Col_name.
from table_name
where Condition 1
    and Condition 2 ;
```

② OR



C_1	C_2	$Y = C_1 + C_2$
F	F	F
F	T	T
T	F	T
T	T	T

Syntax

```
Select * / col_name
from table_name
where Condition 1
    (or) Condition 2 ;
```

Note :- • If we are using 'n' number of Conditions we have to write 'n-1' number of logical operators.
• We should use 'and' operator when all the Conditions have to be satisfied.
• We use 'or' operator when any one of the Condition has to be satisfied.

Ex: WAQTD the Boy details Working as a Software-engineer and also earning more than 1 lakh.

Boys

Bname	Job	Sal
Rahul Gandhi	Comedian	5L
Hucha Venkat	S.E	2.L
Ramesh Jaitchi	S.E	1.5L
Ram	Kabbadi player	4L
Kiran prathap	S.E	0.7L

① Select *
from Boys
where Job = 'S.E'
and Sal > 100000;

↓

F & F	F
T & T	T
T & T	T
F & F	F
T & F	F

② WAQTD the Boy details working either as a S/W engineer (or) earning more than 1 lakh

③ Select *
from Boys.
where Job = 'S.E'
or Sal > 100000;

→

F or F	F
T or T	T
F or F	F
T or T	T
T or F	T

③ WAQTD the emp details working as a Salesman in dept no 30.
④ —u— the emp details reporting to 7802 (or) 7698
⑤ WAQTD the emp names working in dept no 20 or 30.
⑥ WAQTD the emp details for Smith (or) allen.
⑦ —u— ——— working as a manager but earning more than 2000
⑧ —u— ——— working in dept 20 but reporting to 7839
⑨ —u— ——— Working as clerk (or) Salesman
⑩ —u— ——— earning utmost 2000 as an clerk.
⑪ —u— ——— Working as clerk but dept no is 20

↓
means and

(5) Select *
from emp.
where job = 'SALESMAN'
and DEPT NO = 10;

(6) Select *
from emp.
where MGR = 7902
or MGR = 7698;

(7) Select Ename
from emp.
where dept no = 20
(or) dept no = 30;

(8) Select *
from emp
where Ename = 'SMITH'
or Ename = 'ALLEN'

(9) Select *
from emp.
where job = 'CLERK'
and dept no = 20;

(10) WAQTD the employee details earning SAL = 800 (or) 950 (or) 1250

Select *
from emp
where SAL = 800 OR SAL = 950 OR SAL = 1250;

(7) Select *
from emp
where job = 'MGR'
and SAL > 2000;

(8) Select *
from emp
where dept no = 20
and MGR = 7839;

(9) Select Ename
from emp
where job = 'CLERK'
or job = 'SALESMAN';

(10) Select *
from emp
where SAL <= 3000
and Job = 'CLERK';

(08)

(12) Select *
from emp
where job = 'CLERK' and
dept no = 20;

→ IN Operator :- It is a Special Operator used to optimize the query length.

- It avoids multiple usage of OR operator.

Syntax :- LHS in RHS;

LHS in (RHS1, RHS2, RHS3);

↓ multivalued operator.

- It is a multivalued operator (i.e it can handle single LHS and multiple RHS)

Note :- Single valued Operators :- The operators which can take single LHS and single RHS are called single-valued-operators

Ex:- All Relational Operators.

→ NOT IN :-

LHS NOT in RHS;

LHS NOT in (RHS1, RHS2, RHS3);

1) WAQTD the Employee details, ALLEN, WARD & JONES.
2) —11— " " " , who joined on 20-FEB-1981, 19-APR-87
3) —11— Employee names who are earning COM 300, (or) 500 (or) 0.
4) " " Employee details who are not working as CLERK & Salesman
5) " " who are not working in 20 & 30 dept.
6) " " " reporting to 7698, & 7839

1) Select *
from emp
where Ename in ('ALLEN', 'WARD', 'JONES');

2) Select *
from emp
where HIREDATE in ('20-FEB-81', '19-APR-87');

3) Select Ename
from emp.
where COMM in (300, 500, 0);

4) Select *
from emp.
where Job NOT in ('CLERK', 'SALESMAN');

(8) Select *
from emp
where DEPTNO NOT IN (20, 30);

(6) Select *
from emp
where MGR in (7698, 7839);

→ BETWEEN operator :- It is used to Select the data from a given range.
- It is also called as Range operator
- It is also called as Inclusive operator.

Syntax: LHS between LLV and ULV
(NOT)

LLV = lower limit value.
ULV = upper limit value.

- Between operator should be used when the Condition is in the format of Range.

ULV ≥ LLV

→ NOT BETWEEN :-

Syntax :- LHS NOT between LLV and ULV
lower limit value, upper limit value

(1) WAQTD Employee details Earning Salary b/w 2000 to 5000.
(2) ——— " ——— who got hired in year 1981
(3) ——— " ——— Earning COMM with in the range 500 & 1500
(4) ——— " ——— whose Employee No is not b/w 7500 - 7900
(5) ——— " ——— who didn't joined the Company in the year 1980

(1) Select *
from emp
where SAL between 2000 and 5000;

(2) Select *
from emp
where HIREDATE between '01-JAN-81' and '31-DEC-81';

(3) Select *
from emp
where COMM between 500 and 1500;

(4) Select *
from emp
where EMPNO between 7500 and 7900;

⑤ Select *
from emp
where HIREDATE NOT BETWEEN '01-JAN-80' and '31-DEC-80';

Girls

Gname	Age	Boyfriend
Pinchak puja	21	Yes
Disco shanti	50	Yes
Ashika	24	No
Avanthika	26	No
Sunny leone	20	No

NAGTD to give details whose age is b/w 18 to 24 & not having boyfriend

Select *
from emp
where Age between 18 and 24
and Boyfriend = 'No';

→ IS Operator :- It is used to compare with the null value.
There is no other operator in SQL to compare with Null value except IS operator.

Syntax LHS IS NULL;
LHS IS NOT NULL;

① NAGTD who are not earning any COMM
② NAGTD who are not reporting to any MGR
③ —u— reporting to a manager but not earning COMM.
④ —u— earning some COMM but not having a MGR

① Select *
from emp
where COMM is NULL;

② Select *
from emp
where MGR is NULL;

③ Select *
from emp
where MGR > 0 and COMM is NULL;

④ Select *
from emp
where COMM >= 0 and MGR is NULL;
(or)

(or)

(3) Select *
from emp.
where MGR Is NOT Null and COMM is NULL;

(4) Select *
from emp.
where COMM is NOT Null and MGR is NULL;

→ LIKE :- It is used to perform pattern matching and wild-Card Search.

Syntax :- LHS like 'pattern';

- The RHS of the like operator is a pattern
- pattern should be with in Single Quotes.
- like operator can take only 1 pattern at a time.
- pattern is a Combination of Ordinary characters & Meta characters

Meta characters :- [% _]
% → It represents N number of characters ($n \geq 0$)
_ → It represents a Single Character

* Note :- % and _ have this Special functionality only with respect to like operator.

Not like :-

Syntax :- LHS Not like 'pattern';

(1) To AQTD employee names which starts with char-A
(2) —" — which ends with char-N.
(3) —" — designation Value which Contains char-B in it.
(4) —" — designation Value which Contains char-A 2 times in it.
(5) —" — Employee No which starts with 7 & ends with 9.

12. —11— designation value which contains char-R in the last position from last.
13. —11— employee name which contain char L in 2nd position. (Date)

6. AQTO Sal value which contains 2 consecutive 0's in it.
7. —9— the 3 digit SAL value.
8. —11— the employee details who got hired in year 1981 (any date any month)
9. —11— the employee details who got hired in the month of decem-ber (any date any year)
10. —11— employee name which starts with char S or A
11. —11— MGR which don't contain number 8 in it.

Ans

1. Select ename.
from emp.
where like Ename = 'A%';
where Ename like 'A%'; (✓)

2. Select ename
from emp.
where like Ename = '%N';

3. Select JOB
from emp.
where like JOB = '%E%';

4. Select JOB
from emp.
where like JOB = '%A%A%';

5. Select EMPNO
from emp.
where like EMPNO = '7%9';

6. Select SAL
from emp
where like SAL = '%00%';

7. Select SAL
from emp.
where like SAL = '___';

8. Select *
from emp.
where like HIREDATE = '__-___-81';
(or)
'%81';

9. Select *
from emp.
where like HIREDATE = '__-DEC-__';
(or)
'%DEC%';

10. Select Ename.
from emp.
where like Ename = 'S%' OR
like Ename = 'A%';

11. Select * NGR
from emp.
where NOT like NGR = '%8%';

(12) Select JOB
from emp
where like JOB = '%R-';

(13) ~~Select JOB~~
~~from emp~~
~~where like JOB =~~

(14) Select Ename.
from emp.
where like Ename = '-L%';

Pattern - Matching :- If we are using a pattern with the exact position of the ordinary character it is called as pattern matching
Ex:- 'A%';

Wild Card Search :- If a pattern is used without the exact position of the ordinary character it is called as wild card search.
Ex:- '%A%';

→ Escape characters :- It is used to convert Meta characters into ordinary characters.
- The Escape characters are ? ; / , !

① WAQTD Employee names which contains _ in it.?

Select Ename
from Emp
where Ename like '%?_%' escape '?';

② WAQTD designation Value which Contain 2 Consecutive % in it?

Select Job
from Emp
where job like '%?%?%?%' escape '?';

Set Operations

→ SET Operators :- It is used to Combine the output of 2 or more queries.

$q_1 = \{1, 2, 3, 4\}$

Union

$q_2 = \{2, 4, 5, 6\}$.

$q_1 \cup q_2 = \{1, 2, 3, 4, 5, 6\}$

$q_1 \cup_A q_2 = \{1, 2, 3, 4, 2, 4, 5, 6\}$.

$q_1 \cap q_2 = \{2, 4\}$.

$q_1 - q_2 = \{1, 3\}$.

$q_2 - q_1 = \{5, 6\}$.

(U) Union :- It provides both Common & un Common Values without duplicates

Union All ($\cup_A$) :- It provides both Common & un Common Values with duplicates.

Intersect ($\cap$) :- It provides only Common Values.

Minus (-) :- It provides only unCommon Values.

EX:- Select Ename from EMP where JOB = 'SALESMAN' → q_1
union
Select Ename from EMP where DEPTNO = 30; → q_2

q_1	q_1	q_1	q_2
Union All	intersect	Minus	Minus
q_2	q_2	q_2	q_1

→ Concatanation Operator :- It is used to Combine multiple Columns in to a Single Column.

- pipeline operator (||) is used to do Concatanation
- We Can Combine a Column with a Column, String with a Column and also String with a String.

Ex:- Select Ename || '—' || SAL from emp;

o/p

ENAME || '—' || SAL.
SMITH — 800
⋮ ⋮

② Select 'OYE'—' || Ename || '—BYE' From emp;

o/p

OYE SMITH BYE
⋮ ⋮ ⋮

③ Select 'E' || '—SALA' || '—Cup' || '—NANDHE' As RCB From Emp;

RCB.

E SALA CUP NANDHE

Functions :- Function is a block of statements used to perform a specific tasks.

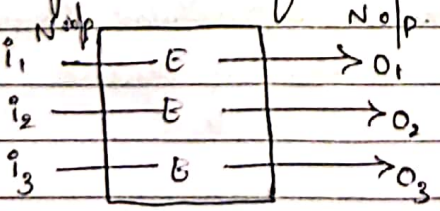
Syntax :- function_name (i/p)

There are 2- types of functions

1) User - defined functions :- The functions which are defined by the user based on the requirement is called as user-defined functions.
   - They can be modified.
   - To develop user - defined functions we need PL/SQL (procedural language Extensions to SQL).

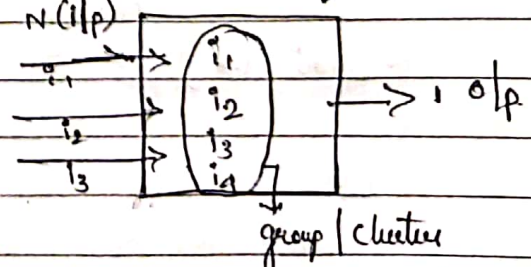
2) pre-defined / built-in functions :- These functions are already defined in the system library.
   - It can't be modified.
   - pre-defined functions are further classified in to 2-types
     a) Single - row - function.
     b) Multi - row - function.

→ Single - row - function (SRF)



- It takes n- number of inputs & provides n- number of output.
- SRF takes single input at a time, executes on that input & Provides a single output.

Multi - row - function (MRF)



- It takes n- number of inputs & provides 1 - output (single output).
- MRF takes multiple inputs at a time, executes by creating a group and provides single output.

SRF

- The output is directly proportional to number of inputs passed.

MRF

- The output is directly proportional to the number of groups created.

* Note :- • Multirow functions are also called as Group-functions (or) Aggregate functions.
• In Multirow functions by default a single-group will be created.

Multi-row-functions

The M-R-F used in SQL are 1) max-function (arg1)
2) min-function (arg1)
3) avg (arg1)
4) Sum (arg1)
5) Count (arg1)

1), 2), 3), 4) → Column name only
5) → Column name / *

1) max (arg1) :- It provides the highest value for the specified inputs.
2) min (arg1) :- It provides the lowest value for the specified inputs.
3) avg (arg1) :- It provides the average value.
4) Sum (arg1) :- It provides the total value.
5) Count (arg1) :- It provides the ~~Number of rows~~ Count the number of values present in a particular column.
• Count (*) - It counts the number of records in the table.

Ex:-

```
Slect max(sal), min(sal), Avg(sal), Count(sal), Count(*), Count(COMM)
```

Max(Sal)	Min(sal)	AVG(SAL)	SUM(SAL)	Count(SAL)	Count(*)	Count(COMM)
5000	800	2073.24	29074	14	14	4

① WAQTD the no of employee working in a dept No 30.
② —11— the highest SAL & lowest SAL among the employee ALLEN, SMITH, WARD
③ —11— the No of employee and total SAL for SALESMAN
④ —11— the average SAL and Total SAL earned by the employee who got hired in 1981
⑤ —11— the earliest the oldest date of joining.
⑥ —11— the highest Commission for the employee reporting to 7698 and 7839.
⑦ —11— the total Sal & No of employee who are not earning any Commision
⑧ —11— the No of employee and the highest SAL for the employee who are not earning SAL in the range 2000-5000.

① Select Count (*)
from emp
where DEPT No = 30;

② Select Max(SAL), Min(SAL)
from emp
where Ename IN ('ALLEN', 'SMITH', 'WARD');

③ Select Count (*), Sum (SAL)
from emp
where JOB = 'SALESMAN';

④ Select Avg(SAL), Sum(SAL)
from emp
where HIREDATE BETWEEN
'01-JAN-81' and '31-DEC-81';

⑤ Select Max(Hiredate), Min (Hire date)
from emp;

⑥ Select Max(Comm)
from emp
where MGR No: In (7698, 7839);

⑦ Select Sum(SAL), Count(*)
from emp
where Comm IS NULL;

⑧ Select Count (*), Max(SAL)
from emp
where SAL Not between
2000 and 5000;

Group-by clause (why) :- To have a combination of column name and a Multi-row-function in the Select statement we use Group-by-clause.

- To create multiple groups in the multi-row-function and to obtain multiple output from the multi-row-function, we use group by clause.

→ • It is used to group the records.

→ Syntax :-

→ Colname + MRF.

④ Select Group-by expression
① from Emp.
② [where <filter condition>]
③ Group by ref_Col
⑤ Order by Ref_Col [ASC] | DESC ;

Ex:

Students

Sname	branch	per.
Basa lingappa	Civil	90
Bonda Ravi	H.M	60
Sexy Seema	Biology	100
Maximuthu	Civil	60
panchayathi parthasthi	Biology	60

→ WAQTD the number of students studying in each branch.

③ Select branch, Count (*)
① from ~~emp~~ students
② group by branch ;

Civil

B.L	CV	90
HM	Civil	60

H.M

B.R	HM	60

Biology

SS	B	100
PP	B	60

final o/p

branch.	Count (*)
Civil	2
H.M	1
Biology	2

① WAQTD the no of employee working in each dept No:
② display the " " working in each job.
③ " " & total salary for the employee reporting to each Manager number.
④ " " dept number wise, No of employee & Highest Salaray for the Employee. who are working in dept No 20 & 30
⑤ " " the job wise least salary except clerk & Salesman.
⑥ " " the No of employee & average Salaray working in each dept No only for the employee who are reporting to 7698, 7839. & sort the output with respect to average SAL in desc order

① Select Count(*), dept No.
from emp
group by dept No;

② Select Job, Count(*)
from emp
group by Job;

③ Select Sum(SAL), Count(*).
from emp
group by MGR;

④ select Count(*), Max(SAL) → dept No.
from emp
where Dept No = 20 AND
Dept No = 30
group by Dept No;

⑤ ~~Select min(SAL), Count(*)~~
~~from emp~~
~~where Job.~~

⑥ Select dept No, Count(*), avg(SAL)
from emp
where mgr in (7698, 7839)
group by dept no
order by avg(SAL) desc;

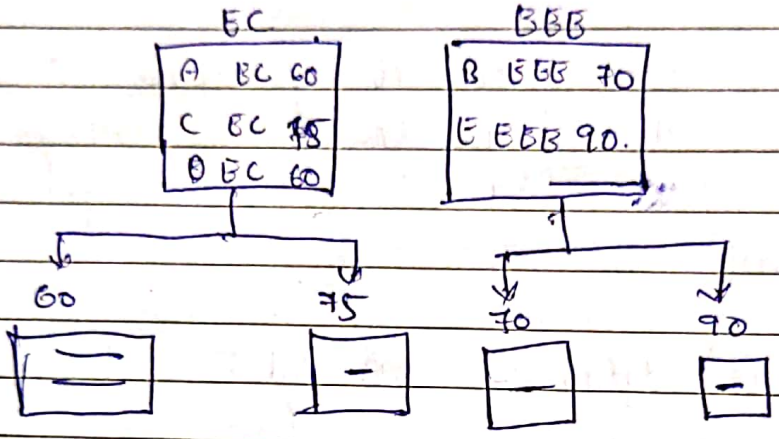
⑧ Select Job, min(Sal)
from emp
where job not in ('CLERK', 'SALESMAN')
group by Job;

students

Sname	branch	%(per)
A	EC	60
B	EEE	70
C	EC	75
D	EC	60
E	EEE	90.

WAP to find the no of students studying in each brach from each per(%).

Select branch, per, Count(*)
from students
group by branch, per;



→ Final o|p

branch	per(%)	Count
EC	60	2
EC	75	1
EEE	70	1
EEE	90	1

→ The Number of groups Created by the group by clause is directly proportional to the Number of unique values present in the reference Column

- It executes in row-by-row format.
- If a Single Column is used in Group-by-clause only Main groups are Created.
- If a Multiple Columns are used in group-by-clause then Main groups along with Sub groups are Created.

① NOAPTD. the no of employees Working in each dept No for each job.

```
Select deptNo, Job, Count(*)
from emp
group by deptNo, Job;
```

→ Rules of Group-by-class.

Rule 1 :- The Columns used in the Select statement [along with AGF] should be specified in the group-by-clause.

Ex:- Select deptno, Count(*)
from emp. → should be same
group by job;

Rule 2 :- The Columns specified in the group by clause may (or) may not be present in the Select statement.

Ex:- Select Count(*)
from emp.
group by deptNo;

Rule 3 :- We Cannot use multirow function in the where clause.

Ex:- Select deptno, Count(*)
from emp.
where Count(*) > 3 (Error Group fn is not allowed here)
group by deptNo;

→ Having clause :- It is used to Specify multirow function or / o Condition.

- It executes after the Group-by-clause.
- It is mainly used to filter the groups.
- Having clause Can't be used without group-by clause.
- It executes in group by group format.

Syntax:

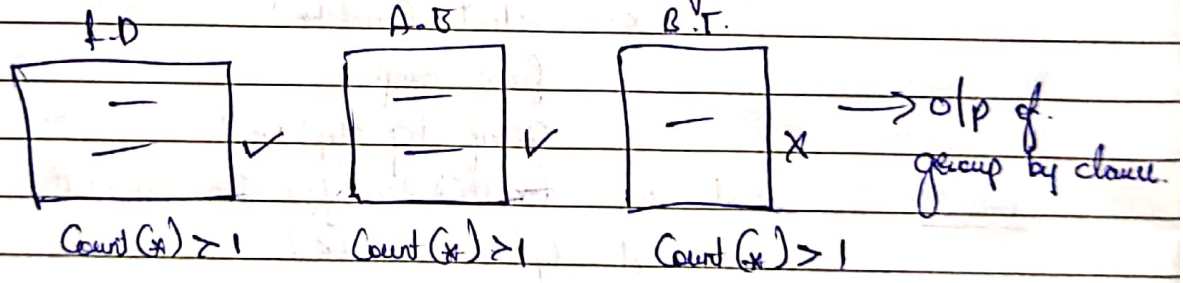
```
Select Group by expression.
from Table name.
[where <filter Condition>]
group by col.
having <Condition>
Order by R.C [ASC]/DESC;
```

Ex: Students

Sname	branch	per
A	f.D	60
B	A.E	70
C	B.T	70
D	A.E	70
E	f.D	70

WAQTD the no of students in each branch only if more than 1 student is studying in the branch.

```
(4) Select branch, Count(*).
(1) from emp.
(2) group by branch.
(3) having Count(*) > 1;
```



Final o/p

branch	Count(*)
f.D	2
A.E	2

(1) WAQTD the No of employee working in each dept if atleast 5 employee are working in that dept.

(2) WAQTD the No of emp for each job only if less than 3 employee are working

(3) WAQTD the no emp reporting to each Manager if at least 4 employees are reporting

④ WAQTD the dept no's wise no of emp only if more than 3 salesman are working in that dept.

⑤ WAQTD job wise total salary only if the total salary is less than 20k for emp working in dept 20 & 30

⑥ WAQTD the no of emp & average salary reporting to each managers no except 7698 only if the avg salary b/w 1000 to 10k.

⑦ WAQTD no of employees with least salary working in each dept only if more than 3 employees are working in the dept & if lowest salary is less than 5000

① select deptno, Count (*)
from emp
group by deptNo
having Count (*) >=5;

② Select job, Count (*)
from emp
group by Job
having Count (*) < 3;

③ Select MGR, Count (*)
from emp
Group by MGR
having Count (*) <=4;

④ Select deptNo, Count (*)
from emp
where job = 'SALESMAN'
group by deptNo
having Count (*) > 3;

⑤ Select Job, Sum (sal)
from emp
where dept No IN (20, 30)
group by Job
having Sum (sal) < 20k

⑥ Select MGR, Count (*), Avg (sal)
from emp
where MGR NOT IN 7698
group by MGR
having avg (sal) between
1000 AND 10000;

⑦ Select dept No, Count (*), min (sal)
from emp
group by dept no
having Count (*) > 3 and
min (sal);

Rules of Having clause :-

Rule 1 :- If a Column name is used as a Condition in the having clause the same must be Specified in the group by clause.

Ex:
```
Select dept no, Count (*)
from emp
group by dept no
having Count (*) > 4 and dept no != 10;
```

Rule 2 :- The Multi row function used in the having clause May or May not be present in the Select statement.

Ex:
```
Select dept no
from emp
group by dept no
having Count (*) > 4;
```

Difference b/w having clause & where clause.

Where Clause	Having clause
• It is used to filter the record	• It is used to filter the groups
• It executes before group by clause	• It executes after group by clause
• Multi-row-function Can't be used	• Multi row function Can be used.
• It Works for un-grouped data	• It Works for grouped data
• It executes in a row-by-row format.	• It executes in a group-by group format.

→ Sub queries :- The query which is used inside the another query is called as a sub query.

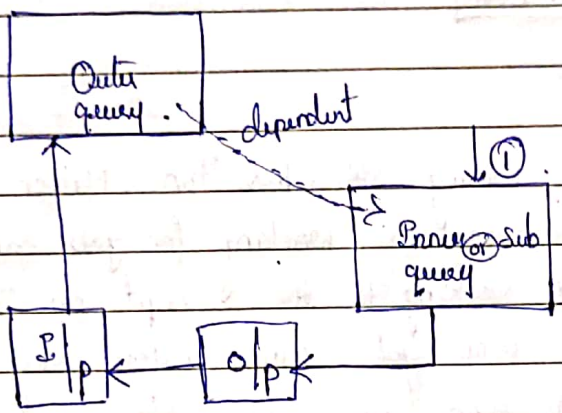
$q_1(q_2)$

q_1 → Outer query
q_2 → Inner query

- Sub queries are also called as Nested queries.
- We can nest up to 255 queries.

$q_1(q_2(q_3(q_4 \ldots q_{255}))$

Working principle :-



- First the Sub query will be executed.
- The output of Sub query is given as an input to the outer query.
- The outer query based on the input received, it will execute and provide the final output.
- Outer query is dependent on inner query but inner query is independent.

Why Sub queries ? (reasons for using Sub-queries.

→ Core :- To find the unknown Values.

Ex 4

Animals

Aname	Colour	Speed
Dog	pink	30
Horse	white	90
cheetah	Yellow	120
Donkey	Grey	10

WAQTD the animals name which runs faster than Dog

```
(1) Select Aname
(2) from Animals
(3) where speed > (Select speed
                   (1) from Animals
                   where Aname = 'Dog');
```

o/p 30

o/p of where

Aname	Colour	Speed
Horse	white	90
cheetha	Yellow	120

final o/p

Aname
Horse
cheetha

outer query

inner query / Sub query

(1) WAQTD employee details earning SAL less than Miller
(2) — " ——— who are working for job same as ALLEN.
(3) — " ——— Names working in the dept no same as SMITH.
(4) — " employee Names who got Hired before king.
(5) — " employee details who are earning SAL less than president.
(6) — " ——— who are earning Commission less than employee id 7654.

(1)

```
Select *
from emp
where SAL < (Select SAL
             from emp
             where Ename = 'MILLER');
```

(01) ID.

(2)

```
Select *
from emp
where job IN (Select Job
              from emp
              where Ename = 'ALLEN');
```

(3)

```
Select Ename
from emp
where DEPT NO IN (Select dept No.
                  from emp
                  where Ename = 'SMITH');
```

(4)

```
Select Ename
from emp
where HIREdate < (Select HIREdate
                  from emp
                  where Ename = 'king');
```

(5) Select *
from emp
where SAL < (select SAL
from emp
where JOB = 'president');

(6) Select *
from emp
where COMM < (Select COMM
from emp
where EMPNO = 7654);

(7) WAQTD the Eployee details. who are working in dept No SAME AS SMITH & ALLEN.

Select *
from emp
where DeptNo IN (Select DeptNo.
from emp
where Ename IN ('SMITH', 'ALLEN'));

(8) WAQTD the Employee details who are not reporting to SMITH & ALLEN MGR No

Slect *
from emp
where MGR No Not IN (Select MGR No
from emp
where ENAME IN 'SMITH', 'ALLEN'));

⟹ Type of Sub query.

① Single-row-Subquery :- if the sub-query is providing a single o/p it is called as Single-row-subquery.

⟹ To Compare with Single-row subquery we can use IN & = operator.

⟹ (relational operator).

② Multi row Subquery :- If the Subquery is providing more than 1 output it is called as Multi-row-Subquery.

- To Compare with Multi-row-Subquery, we can only use IN & NOT IN operators.

→ 'ANY' and 'ALL' operator :-

These operator should be used only with relational operator.

- ANY operator will select the lowest value from the Subquery output
- All operator will select the highest value from the Sub-query output.

Ex: WAQTD the emp details earning Sal More than atleast a Salesman

```
Select *
from emp
where SAL > ANY (Select sal
                 from emp
                 where Job = 'SALESMAN');
```

① WAQTD the emp details earning Sal More than all the Salesman.

```
Select *
from emp
where SAL > ALL (Select sal
                 from emp
                 where Job = 'SALESMAN');
```

Case ② :- Mapping two (or) more tables by using Common Columns b/w them.

- We can Map the tables only if they have Common Columns i.e (Relation ship).

EX:- WAQTD the movie name which is running in Navarang theater.

Movies

Mname	Mid
Nuvvu dese	1
OM	2
A	3

Theaters

Mid	Tname
1	Navarang
2	Prasanna
3	Vravathi

```
Select Mname
from Movies
where Mid in (Select Mid
              from theaters
              where theaters = 'Navarang');
```

(or)

```
Select Mname                          Mname
from Movies;                          Movies
where Mid in (Select Mid              Mid
              from theaters           theaters
              where theaters = 'Navarang');   Tname.
```

- WAQTD theaters name where OM Movie is running

```
Select tname.
from theaters
where Mid in (Select Mid
              from Movies.
              where Mname = 'OM');
```

Note :- If the data to be fetched is in one table and the Condition is in another table we go for Case ②

① WAQTD the emp name working in accounting dept
② WAQTD the dept name for the emp allen & SMITH.
③ WAQTD the emp details working in newyork & Chikago.
④ WAQTD the deptname & location for the emp working as Salesman.
⑤ WAQTD the dept details for the emp's who gained the Comp in year 1981.
⑥ WAQTD the dept emp name & Salary for the employees who are not Working in Sales dept.
⑦ WAQTD the dept name for the emp who are earning Sal in range 2000 to 5000

① Select ename
from emp.
where dept no in (Select dept no
from dept
where ~~dname IN ('ALLEN','SMITH');~~
dname = 'ACCOUNTING');

② Select the dname.
from dept
where dept no In (Select dept no
from ~~dept~~ emp.
where ename IN ('ALLEN','SMITH'));

③ Select *
from emp.
where dept no In (Select dept no.
from dept
where loc IN ('NEWYORK','CHICAGO'));

④ Select dname, loc.
from dept
where deptNo IN (Select deptNo
from emp
where job IN 'SALESMAN');

⑤ Select *
from dept
where deptNo IN (Select deptNo.
from emp.
where HIREDATE LIKE '%-%-81');

⑥ Select ename, SAL
from emp
where deptNo IN (Select deptNo.
from emp
where ~~Deptno~~ Dname NOT IN 'SALES');

⑦ Select dname.
from dept
where deptNo IN (Select deptNo.
from emp.
where SAL between 2000 AND 500);

Case 3: To find n^{th} Maximum (or) n^{th} Minimum Value

2nd MAX < 1st MAX 2nd MIN > 1st MIN.

emp

ename	Sal.	
A	500	✗
B	200	✓
C	100	✓
D	400	✓
E	300	✓

① WAPTO the 2nd Maximum Salary

Select max(sal)
from emp (500)
where sal < (Select max sal
from emp);

final o/p
400

2) WAQTD the 2nd Minimum Salary.

```
Select min(sal).
from emp                    (100)
where sal > (Select min(sal)
             from emp);
```

final o/p 200

3) WAQTD the 3rd Minimum Salary.

```
Select min(Sal) from emp
where sal > (Select min(sal) from emp
where sal > (Select min(sal) from emp));
```

4) WAQTD the 3rd Maximum Salary.

```
Select Max(sal) from emp.
where sal < (Select max(sal) from emp
where sal < (Select max(sal) from emp));
```

1) WAQTD the 2nd Minimum COMM
2) WAQTD the 4th recent date of joining
3) — " — the first Maximum Sal & 2nd Maximum Sal.
4) — " — the Second Maximum & 2nd Minimum Sal.

1)
```
Select Min(COMM) from emp
where Comm > (Select Min(COMM) from emp.
where Comm > (Select Min(Comm) from emp));
```

2)
```
Select Max(Hiredate) from emp.
where Hiredate < (Select Max(Hiredate) from emp
where Hire date < (Select Max(Hiredate) from emp.
where Hire date < (Select Max(Hiredate) from emp)));
```

③ Select Max(Sal) from emp
Union
Select Max(Sal) from emp
where Sal < (Select Max(Sal) from emp);

④ Select Max(Sal), Min(Sal) from emp.
where Sal < (Select max(Sal) AND Sal > select min(Sal)
from emp);

⑤ Select Max(Sal) from emp
where Sal < (Select Max(Sal) from emp);
Union.
Select Min(Sal) from emp.
where Sal > (Select Min(Sal) from emp);

Note:- To find n^{th} Maximum (or) Minimum Value we should write n number of queries.

Case-4 :- Mapping a table to itself by using different Columns of the table.

emp

emp no	Ename	mgr
1	Naveen	3
2	Sumanth	1
3	Sowjanya	2

Naveen reporting Sowjanya
↓ ↓
Subordinate Manager

① WAQTD the manager name for the Employee naveen.

Select ename
from emp.
where emp no in (Select mgr
from emp
where ename = 'Naveen');

o/l o/p
ename
Sowjanya

② WAQTD Subordinate name for the reporting to Naveen.

```
Select ename
from emp
where mgr in (Select empno
              from emp
              where ename = 'Naveen');
```

Hint

	Subordinate name	Manager name
Subquery Select Statement	Employee no	Mgr.

① WAQTD for the MGR name for employee Smith.

```
Select ename
from emp
where empNo in (Select MGR no
                from emp
                where ename = 'SMITH');
```

② Display the Subordinate name reporting to ALLEN KING.

```
Select ename
from emp
where MGR in (Select empNo
              from emp
              where ename = 'KING');
```

③ Display the Manager name for the employee ALLEN & SMITH

④ Display the Subordinate name who are not reporting to SMITH & all

③ Select ename.
from emp.
where emp no In (Select MGr
from emp
where ename In ('ALLEN','SMITH'));

④ Select ename
from emp
where MGr not In (Select emp no
from emp
where ename In ('SMITH','ALLEN'));

Drawbacks of Sub-Query.

- The output is valid but not clear. (Case-2)
- In Case-2 we can map Multiple table but we can't fetch Multiple table data.
- In Case-3 to find n^{th} Maximum (or) n^{th} minimum value we have to write n number of queries, this makes the query lengthy, Time taken for the execution will be more, therefore performance will be poor / not good.
- In Case-4 We can't fetch both Subordinate name & Manager name at the same time.

⁂ → **JOINS** :- Joins are the special posibility of the Select statement used to fetch data from Multiple tables simultaneously.
- By using joins we can also fetch single table data.
- There are 5 types of joins.
  1) Cartesian join
  2) Inner join / Equi join
  3) Non - Equi join
  4) outer join
  5) Self - join.

→ 1) Cartesian Join :- It is a Cross-product of 2-(or) more tables.
- It Works based on Cartesian principle.
- According to Cartesian principle each & every record of one table will be merged with all the records of other table.
- Cartesian join doesn't use joining Condition [No where clause]
- It provided both valid & invalid records in the output.
- The Count of invalid records will be more than Count of valid records and this is the main drawbacks of Cartesian join.

Oracle Syntax :-

```
Select * / Col_name
from table name 1, table name 2;
```

ANSI Syntax

```
Select * | Col_name.
from table name 1 Cross join table name 2;
```

Drinks.

Dname	itemno
OT	1
old monks	2
RPM	3

bar

itemno	bname
1	Nawarang bar
2	Ranga bar
3	Trishul bar.

① WAQTD the drink name & bar name for all the drinks

```
② Select dname, bdname
① from drinks, bar;
```

ANSI query
Select dname, bname
from drinks Cross join
bar;

o/p of from clause

Dname	item no	item no	bname
OT	1	1	N.B
OT	1	2	R.B
OT	1	3	T.B
old monks	2	1	N.B
old monks	2	2	R.B
⋮	⋮	⋮	⋮

invalid records = 6
Valid records = 3
∴ invalid > Valid records

final o/p.

dname	bname	
OT	NB	✓
OT	RB	✗
OT	TB	✗
old monks	NB	✗
old monks	RB	✓
old monk	TB	✗
⋮	⋮	✗
⋮	⋮	✓

(2) WAQTD. Emp name & dept name for all the employees!

oracle :- Select ename, dname
from emp, dept;

ANSI :- Select ename, dname
from emp cross join dept;

Note :- Cartesian join can be applied for the tables with are without relationship.

2) Inner JOIN :- / EQUS JOIN :- Joining Multiple tables by using Common Columns between the tables is called Inner join (or) Joining Multiple tables by specifying a joining condition is called as Inner join.

- Inner join can be applied for the tables only if they have relationship. (Common Columns)
- It provides only valid records in the output.

Oracle - Syntax :-
Select * / Column - name
from table name 1, table name 2
where T1.CC = T2.CC. (CC = Common Column)
↓ joining condition.

ANSI Syntax :-
Select * / Column - name.
from T1 [INNER] JOIN T2
ON T1.CC = T2.CC.
↓ joining condition

→ for joining 3 tables
Select * / Column name.
from table 1, table 2, table 3
where T1.CC = T2.CC and T2.CC = T3.CC
↓ joining condition.

ANSI
Select * / Column - name.
from T1 [INNER] JOIN T2.
ON T1.CC = T2.CC [INNER] JOIN T3
ON T2.CC = T3.CC.

Ex:- drinks

dname	item no.
Blackdog	1
M.H	2
Royal stag	3

bar.

Item no	bar name.
1	Naveen Wines
2	Adarsh wines
3	Nirmal Wines

WAQTD the drink name & bar name for all drinks.

oracle:- Select dname, bname
from drinks, bar.
where drinks.itemno = bar.itemno;

Ansi:- Select dname, bname
from drinks (join) bar
ON drinks.itemno = bar.itemno;

(or)

Select dname. bname
from drinks a, bar b.
where a.itemno = b.itemno;

Final op.

dname	Bname
Blackdog	Naveen Wines
MH	Adarsh Wines
Royal stag	Nirmal Wines

* ⇒ Joining Condition :- It is a special condition which joins 2-tables

- These join is also called as equi join because we use equal operators and cc. where in the joining condition.
The format for the joining condition is [T1.cc = T2.cc].

Q.1) WAQTD the emp name & dept name for all the employee.

Select ename, dname
from emp, dept
where emp.deptno = dept.deptno;

(or)

Select ename, dname
from emp join dept
ON emp.deptno = dept.deptno;

(or)

Select ename, dname.
from emp a, dept b
where a.deptno = b.deptno;

Q.2) WAQTD the emp name, location & Salary for all employees

Q.3) — " — dept name for employee who got hired in year 1981

Q.4) — " — emp name, Comm & location for the employee who are working as Manager in research dept.

Q.5) — " — dept name for all the employees who are earning Same Comm.

Q.6) — " — the emp no, emp name & dept name for the employee who are not working in chikago.

Q.7) — " — the emp details & dept name for the employee who are reporting to 7839

Q.8) — " — the emp details & dept details for the emp working in research & Sales department.

Q.9) — " — the emp name, emp details & location for the emp's who are earning 4-digit Salary.

Q.10) — " — the emps dept no, a department dept no for the emp's who are not earning Salary b/w (500 to 2000).

Q.11) W.A.Q.T.D the region name and Country name for all the Countries

```
Select rname, Cname
from regions, Countries
where regions.id = Countries.cid;
```

1) Customers

Cid	Cname

products

Pid	Pname	Cid

Food

item no	fname

Hotel

Item no	Hname

Q12) W.A.Q.T.D the Customer name & product name for all the products

```
Select Cname, Pname
from Customers a, products b
where a.cid = b.cid;
```

PAGE NO :
DATE : / /

→ Left outer join

Syntax

oracle.

```
Select * /Col(C)
from T1, T2
where T1.CC = T2.CC(+);
```

Ans:

```
Select * /Col(C)
from T1 left join T2
ON T1.CC = T2.CC;
```

→ Right outer join

oracle

```
Select * /Col(C)
from T1, T2
where T1.CC(+) = T2.CC;
```

Ans:

```
Select * /Col(C)
from T1 Right join T2
ON T1.CC = T2.CC;
```

→ full outer join

oracle.

```
Select * /Col(C)
from T1, T2
where T1.CC = T2.CC(+);
union
Select * /Col(C)
from T1, T2
where T1.CC(+) = T2.CC;
```

Ans:

```
Select * /Col(C).
from T1 full join T2
ON T1.CC = T2.CC;
```

① WAPD the employe name and dept name for all the employee and also display the employee name who is not working in any dept

oracle

```
Select ename, Dname
from emp, dept
where emp.deptno = dept.deptno;
Union
select ename
from emp
where deptno is Null;
```

(or)

```
Select ename, Dname
from emp, dept
where emp.deptno = dept.
deptno(+);
```

(esir)

PAGE NO :
DATE : / /

ANSI

```
Select ename, Dname
from emp left join dept
ON emp.deptno = dept.deptno;
```

→ Display the employee name & deptname for all the employee & also display dept name where no employee are working

Oracle

```
Select ename, dname
from emp, dept
where emp.deptno(+) = dept.deptno;
```

ANSI

```
Select ename, dname
from emp Right join dept
ON emp.deptno = dept.deptno;
```

→ Display the emp name & dept name for all the employee & display the emp name whose is not working in any dept and also display dept name where nobody is working.

Oracle

```
Select ename, dname
from emp, dept
where emp.deptno = dept.deptno(+);
union
Select ename, dname
from emp, dept
where emp.deptno(+) = dept.deptno;
```

ANSI

```
Select ename, dname
from emp full join dept
ON emp.deptno = dept.deptno;
```

Self Join :- joining a table to itself is called as self join.

Case 1 :- joining a table to itself by using different columns of the table.

The joining condition is $T_1.C_1 = T_1.C_2$

A

erp

PAGE NO :
DATE : / /

Ex:

Emp A

empno	ename	mgr
1	Ashwin	2
2	Mamtha	3
3	Jayanth.	1

Emp B

empno	ename	mgr
1	Ashwin	2
2	Mamtha	3
3	Jayanth.	1

③ Select A.ename as Sub, B.ename as mgr.
① from emp A, emp B.
② where A.mgr = B.empno;

final output:

Sub	Mgr
Ashwin	Mamtha
Mamtha	Jayanth.
Jayanth	Ashwin

① WAQTD the Subordinate name & Mgr name for the employees only if the Subordinate are earning salaray less than their Mgrs.

Select A.ename as Sub, B.ename as Mgr.
from emp A, emp B.
where A.mgr = B.empno and A.Sal < B.Sal;

② WAQTD. the Mgr name of the employee SMITH
③ Display the Subordinate name reporting to Allen
④ —u— the —u— & Manager name for all the employees only if the managers got hired after their Subordinate.

② Select A.ename as Sub, B.ename as Mgr
from emp B, emp B
where A.ename = 'SMITH';
A.mgr = B.empno

③ Select A.ename as Sub, B.ename as Mgr
from emp A, emp B
where B.ename = 'ALLEN';
A.mgr = B.empno

PAGE NO :
DATE : / /

⑤ Select A.ename as Sub, B.ename as MGr.
from emp A, emp B
where A.mgr = B.empno and A.Hiredate < B.Hiredate;

Case-2 :- Joining a table to itself by using Same Columns of the table.

- Joining Condition is $T_1.C_1 = T_1.C_1$.

① WAQTD the employee names who are earning same sal.

Select A.ename, B.ename
from emp A, emp B
where ~~A.deptno = B.deptno and~~ A.Sal = B.Sal And A.Ename ≠ B.Ename;

② WAQTD the employee names who Got hired on same date

Select A.ename, B.ename
from emp A, Emp B
where A.Hiredate = B.Hiredate and A.Ename ≠ B.ename;

Q13) W.A.Q.T.D the food name & hotel name for all the food's.

```
Select fname, hname
from food a, hotel b
where a.item no = b.item no;
```

3 tables

Q.14) W.A.Q.T.D the region name, Country name & City name for all the Cities.

```
Select region_name, Country_name, City_name
from regions a, Countries b, locations c.
where a.Region_id = b.region_id and b.Country_id = c.Country_id;
```

Note :- If we are joining n number of tables we have to Specify (n-1) no of joining Conditions.

→ Non-Equi-join :- Joining Multiple tables by without using Common Columns is Called as Non-Equi joins.

- Joining Multiple tables without Specifying equals operator in the joining Condition is Called as Non-Equi joins.
- To apply Non-equi join 2-Conditions has to be Satisfied.
- The tables to be joined should not have Common Columns [No relationship]
- The tables to be joined should have relevant data blw them.

→ Relevant data :- The data between the tables which are having same data type and size irrespective of the Column names is Called as relevant data.

Q15) Display Empname, Salary & Grade for all the employees.

```
Select Ename, Sal, Grade.
from emp, salgrade.
where Sal BETWEEN losal and hisal;
```

Outer joins :-

Right table.

Note :- :- Where T1.cc = T2.cc.

Left table.

Joining Multiple tables by using Common Columns b/w them is called as outer join.

- Outer joins provides valid records from both left & Right tables and invalid records from left (or) right (or) both left and right table.

Outer join = Valid records (left & Right table) + invalid (left (or) right table)

Outer joins is nothing but = Inner join + invalid (left (or) right) table

There are 2-types of outer joins

1) Left outer join = Inner join + invalid (left) table
2) Right outer join = Inner join + invalid (Right) table.
3) Full outer join = Inner join + invalid (left & Right) table.

Syntax

oracle

```
Select * / Cols
from T1, T2.
where T1.cc = T2.cc(+);
```

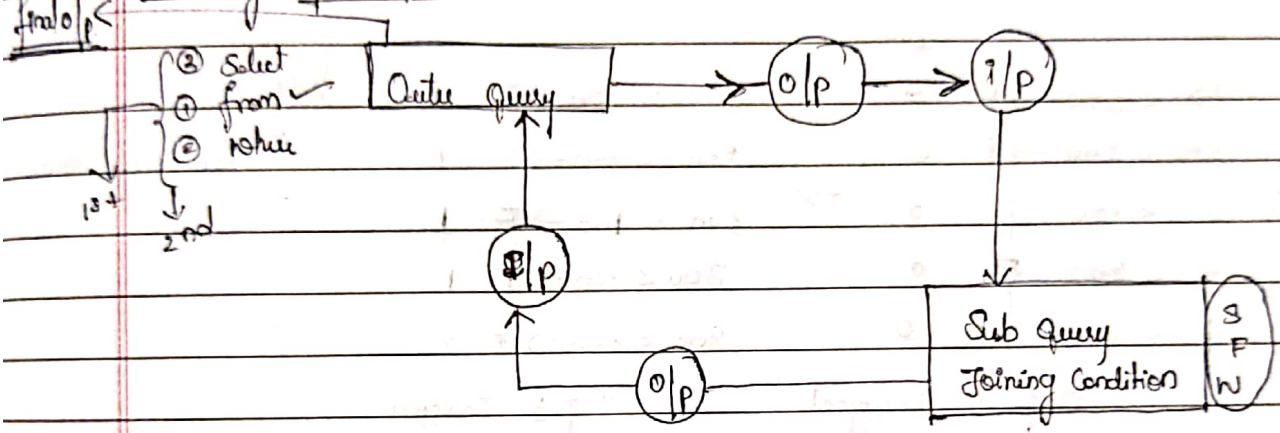
Co-Related Sub-Query

It is an advanced type of sub-query, there sub query is having a joining condition

→ Why Co-related Sub-queries :-

- To find n^{th} maximum & n^{th} minimum value with optimization.

Working principle :-



- First sub query will be executed, due to the presence of the joining condition outer query will execute first.
- Outer query for the first time executes partially (only from clause) and provides the output, this output is given as an input to sub query.
- Sub query executes completely and provides the output, this output is given as an input to the outer query.
- Outer query based on the input received executes completely (select clause, from & where clause) for the second time and provides the final output.
- Here outer query and inner query are interdependent.

classmate
Date
Page
To remove the duplicates

Emp A

Ename	Sal
A	500
B	800
C	100
D	200
E	400

Emp B

Ename	Sal
A	500
B	800
C	100
D	200
E	400

→ WAQTD the 3rd max Salary.

Select distinct sal
① from emp A
where 3 = (Select Count (distinct sal)
① from emp B
② where A.sal < B.sal);

(n-1)
when nth max sal/min sal is req

	Count
500 < 500 → F	0
500 < 300 → F	0
300 < 100 → F	0
300 < 200 → F	0
300 < 400 → T	0

0 ≠ 3 so reject

	Count
300 < 500 → T	1
300 < 300 → F	1
300 < 800 → T	2
300 < 200 → F	1
300 < 400 → T	2

2 ≠ 3 so reject

Why for all Values.

200 < 500 → T 1
200 < 300 → T 2
200 < 100 → F 2
200 < 200 → F 2
200 < 400 → T 3
3 = 3 accept.

final output = 200

Sal
200

① WAQTD the 4th minimum Salary
② —11— 8th maximum Salary
③ —11— 6th minimum Salary.

Normalization

It is a process of decomposition of large table in to Several small tables (or) it is a table design technique which Organizes the data by avoiding Redundancy and dependency.

Redundancy :- It is unwanted repetition of data

Normalization was introduced by E.F Codd in the years '1970'
E.F Codd introduced Several normal forms.
The Normal form's are.

→ 1 NF (1st Normal form)
→ 2 NF (2nd —u—).
→ 3 NF
→ 3.5 NF (B.C.N.F) → Became invented by both Raymond Boyce E.F Codd.
→ 4 NF
→ 5 NF
→ 6 NF

Note :- A table is identified as Simple (or) Complex by applying Normal form's.

→ 1 NF Rules :-

Rule 1 :- Each and Every Cell of Table Should Store Single Value.

Rule 2 :- All Column names in the table must be unique.

Rule 3 :- Duplicate rows should not be present in a table.

→ A table is said to be Complex if it is storing more than one Value is any of its cells.

Shirts

PK

Ex:1

Sid	Brand	Size	Colour
1	Adidas	S	Black
2	levis	L	Blue
3	Jockey	XL	white, PINK
4	RamRaj	XXL	white

Complex

Shirts

Sid	Brand	Size	Colour
1	A	S	Black
2	L	L	Blue
3	J	XL	White
3.1	J	XL	PINK
4	RamRaj	XXL	white

Simplified to INF.

Ex:2

emp

Cid	Cname	Sal
1	Naveen	300
2	Adarsh	301.2
3	Sanjanya	55
4	pooja	8888
5	Achutha	400

Complex

emp

Cid	cname	Sal	bonus
1	N	300	
2	Ad	301	2
3	S	55	
4	P	8888	
5	Ac	400	

Simplified to INF.

⇒ 2NF (Second Normal Form).

Rule 1 :- The table should exist in first normal form.

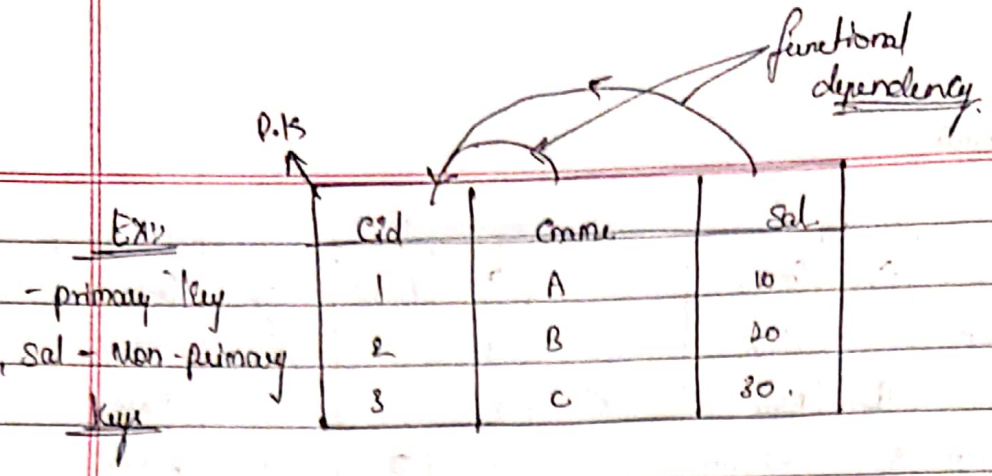
Rule 2 :- Each and every table should have a primary key column.

Rule 3 :- Functional dependency should exist.

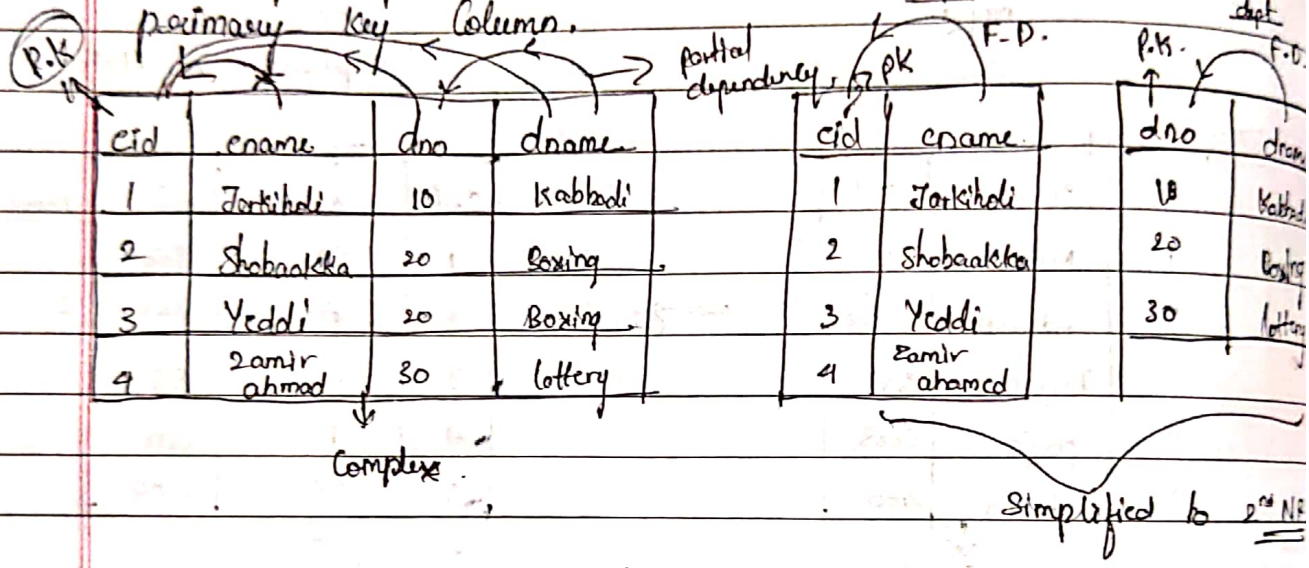
Rule 4 :- partial dependency shouldn't exist.

⇒ A table is said to be Complex if it is not having a primary key & if it has partial dependency

Functional dependency :- It is a kind of dependency where all the Non-primary key Columns of a table will be dependent on the primary key Column of the table.



Partial dependency :- It is a kind of dependency where a primary key Columns will be dependent on another primary key Column.



→ 3NF (Third Normal Form)

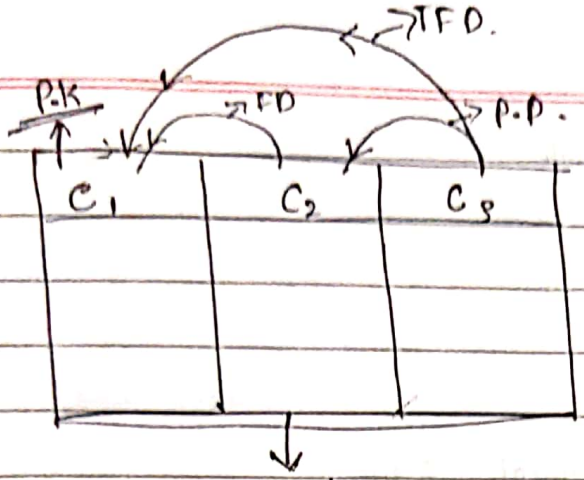
Rule 1:- Table should exist in 2nd Normal form.

Rule 2:- Transitive functional dependency shouldn't exist.

→ When Table is said to be Complex if it Contains Transitive functional dependency (TFD).

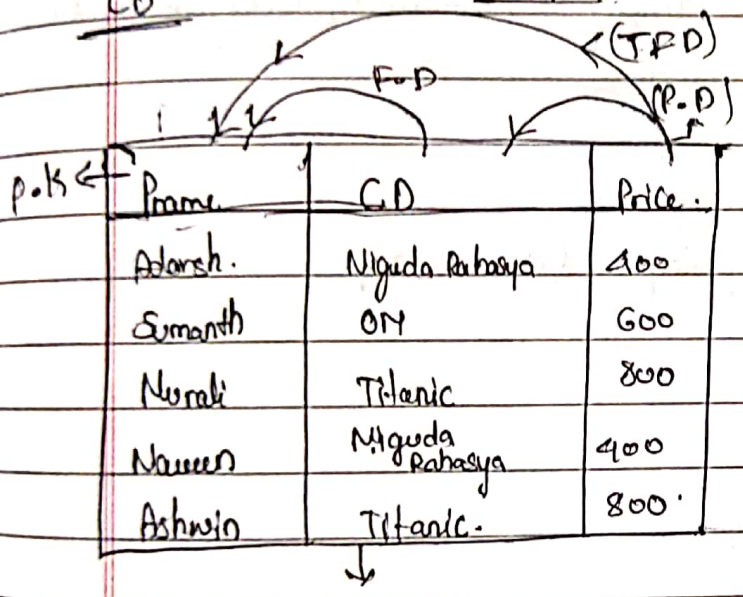
TFD :- Consider 3 columns C_1, C_2, & C_3, if C_3 is dependent on C_2 and C_2 is dependent C_1 then indirectly C_3 will be dependent on C_1, This kind of dependency is called as Transitive functional dependency (TFD).

Ex:



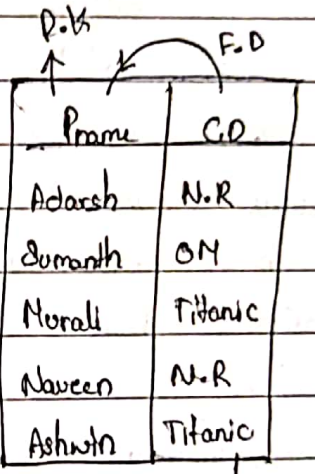
Complex

CD

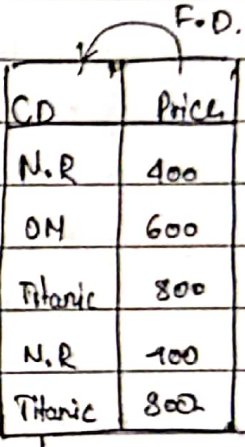


Complex.

CD-1



CD-2



Common column (Relationship).

Simplified to 3rd NF

Resume Points

SQL interview Questions

- SQL Sub language.
- Diff b/w delete Truncate, Drop
- Joins types → Cartesian join, Inner join (queries), outer join (syntax)
  Self join & Non-equi join.
- Diff b/w DBMS & R-DBMS.
- Diff b/w primary key & Foreign key
- Subquries (Cases) & Working principle (Case-3 imp)
- Diff b/w Constraints
- Normalization (Rules)
- Group by & having clause.
- Co-related Subquries, Special operators.