SQL → Structured Query Language
↳ Data Base
↳ Data

→ Select * from Cat; → Displays Table Name

→ desc Table Name;

→ Select * from Table Name;

Columns → Structured Schema/ desgr

Row → Individual Data

Data Base :-

Data Base is a Collection of data in a format that can be easily accessed. (Digital)

A Software application used to manage our DB is called DBMS (Data Base Management System).

Types of Data Base :-

Relational → SQL	Non-Relational
Data Stored in tables	Data not Stored in tables.

→ SQL is a Programming language used to interact with Relational Data Bases.

It is used to perform CRUD operations:

Create
Read
Update
Delete.

Creating our first Data Base :-

Our first SQL Query

→ Create DataBase db_name;

Ex:- Create Data Base Vijay;

→ Drop Data Base db_name; → Deletes the data

Ex:- Drop data Base Vijay;

Creating Our First Table :

Use db_name; → It Creates data in which we Created a new data base.

```
CREATE TABLE table_name (
    Column_name1 datatype Constraint,
    Column_name2 datatype Constraint,
    Column_name3 datatype Constraint
);
```

College

Use db_name;

Ex:-

```
CREATE TABLE Student (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT NOT NULL
);
```

id → INT → Because we use integers that's why we use int key word

name → VARCHAR(80), ↓ we use alphabets that why we use VARCHAR keyword with min (50) char.

age → INT NOTNULL ↓ NOTNULL means age section should not be empty.

```
INSERT INTO Student VALUES (1, "Vijay", 22);
INSERT INTO Student VALUES (2, "Durga", 22);
SELECT * FROM Student
```

DataTypes:-

- → CHAR
- → VARCHAR
- → BLOB
- → INT
- → TINYINT
- → BIGINT
- → BIT
- → FLOAT
- → DOUBLE
- → BOOLEAN
- → DATE
- → YEAR

SQL DataTypes:

TINYINT UNSIGNED (0 to 255)

TINYINT (-128 to 127)

Types of SQL Commands:

DDL - (Data Definition language): Create, alter, rename, truncate & drop

DQL - (Data Query language) : Select

DML - (Data Manipulation language): Insert, Update & delete

DCL - (Data Control language): grant & revoke permission to users

TCL - (Transaction Control language): Start transaction, Commit, Rollback

Data:- Data is a useful information & all the data are stored in a centralized location called "Data Base".

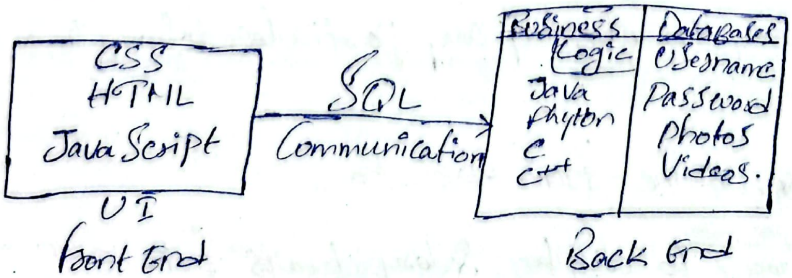
Data Base:- It is defined as a useful information which is organised by following certain set of rules is called "DB".

DBMS:- It is defined as a collection of programs which is written in order to manage the Data Base effectively.
Edgar F. Codd 1970 i-DBMS
Here managing stands for - Storing a data
Fetching the data
Updating & deleting the data.

Types of DBMS:-
→ Flat file DBMS
→ Hierarchical DBMS
→ Network DBMS
→ RDMS.

RDBMS:- It is defined as a collection of data which is organised by following some relational rules is called an "RDBMS".
In this type data will be stored in the form of "tables". (Table contain Rows & Column). We can create 'n' no. of tables & we can build relationship b/w them. to fetch the data.

SQL:- Raymond Boyce, Donald Chamberlin } IBM Researchers in 1970

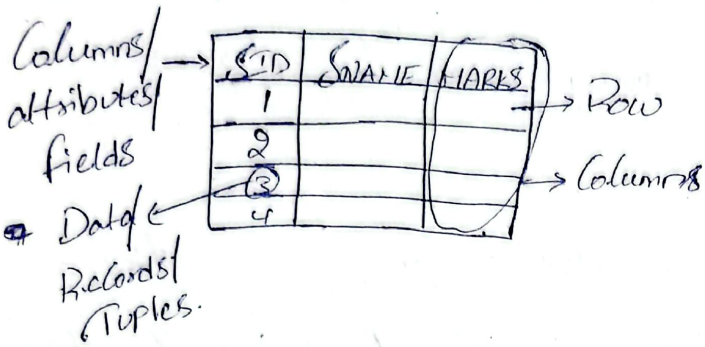


Whenever User want to Communicate with the "DB" since using SQL we can Communicate with the "DB"

SQL is a high level language (Human understandable language).

SQL is not a case sensitive but the data inside the table is case-sensitive.

Every Query must end with a Semi Colon (;).



54

Entity:- Each & Every Tables we can call it as "Entity".

Ex:- 

Attributes:- The characteristics of a table is called "Attributes".

→ To Display the Table Names present in any database then we have to use the below Query.

* Select * from Cat; (or) Tab;

→ To Display Column names from particular table then we have to use below Query.

* desc table name;

→ To Display particular table present in the Data Base then we have to use the below Query.

* Select * from Table Name;

→ Write a Query to display DepartName?

* Select * from Dept;

→ Write a Query to display any particular column in a table Name?

* Select Column Name from Table Name;

→ Write a Query to display Salary details from EMP table.

* Select SAL from EMP;

→ Write a Query to display designation details from EMP?

* Select Job from EMP;

→ Write a Query to display employ name & their joining date from their EMP table:

* Select EName, Hiredate from EMP;

→ Write a Query to display ENames, Salary & their Commission from EMP Table.

* Select EName, SAL, COMM from EMP;

Note: Table Names & Column Names need to be used as it is present in the table.

Knowing the table Name & Column Name is mandatory.

Aliasing

→ Giving another Name for existing table (or) a Column Name

→ To achieve aliasing "as" keyword.

→ Syntax for aliasing a Column Name.

Syntax: Select ColumnName as aliasname
from TableName;

Ex: Select Sal as Salary from EMP;

→ Aliasing is not a permanent change it is only for display purpose.

→ We can achieve aliasing without using "as" keyword.

Syntax: Select Columnname aliasName from TableName;

Ex: Select SAL Salary from EMP;

→ When you want to display a name as it is given in the Query we have to use Double Quote (" ")

Syntax: Select Columnname "aliasname" from table name;

Ex: Select SAL "Salary" from EMP;

→ Syntax for aliasing a tablename → aliasname

Syntax:- Select Columnname from tablename aliasname;

Ex:- Select SAL from EMP Employee;

Arithmetic Operators

→ Plus (or) Addition '+'
→ Subtraction '-'
→ Multiplication '*'
→ Division '÷' → /
→ Modulus 'mod'

(*) On Select Statement we can use arithmetic operators.

→ Write a Query to Display all the employNames & increasing the Salary of 100 RS.

Ex:- Select EName, Sal+100 from EMP;

→ Write a Query to Display EName, [illegible] & Decreasing the Salary 500 RS. from EMP table.

Ex:- Select EName, '[illegible]', SAL-500 from EMP;

→ Write a Query to Display all the ENames & their annual Salary.

Ex:- Select EName, SAL*12 from EMP;

→ Write a Query to Display all the [illegible] ENO & their daily Salary.

Ex:- Select ENO, SAL*12/365 from EMP;

→ Write a Query to Display EName & increasing the Salary of 2%.

Ex:- Select EName, SAL+(Sal*2/100) from emp;

→ Write a Query to Jobs & their Salary divisible by 2.

Ex:- Select Job, /Sal/, 2; => Select Job, mod(Sal,2) from Emp;

→ Write a Query to display dept no & their Salary which is divisible by 2.6.

Ex:- Select dept no, Mod (Sal, 2.6) from Emp;

Literals:- Usage of data directly into a Query is known as "Literals" we have three types of Literals → String, Number, Date

(*) we have to use String & date data inside a Single quotes. & we can Number data directly into a Query.

Dual:- It is a dummy table which Contains one row & one Column.

Ex:- Select * from dual.

When we want to access a data randomly which is not inside a data base then we will use dual table.

→ Select 4578 from Dual;

4578
4578

→ Select 'SQL' from Dual;

'SQ
SQL

→ Select '05-02-2001' from Dual;

'05-02-200
05-02-2001

Concatenation

Joining (or) Merging two (or) More Columns (or) literals is Called as "Concatenation".

To achieve Concatenation we have to use double pipe Symbol → "||".

Ex:- Select ename || Job from emp;

(*) Select 'myname is' || 'divya' from dual;

'MYNAME is' || 'D
myname is Divya

(*) Select 'my name is' ||ename from emp;

→ Write a Query to display in the below format.

→ MyName is SMITH

→ My Name is WARD

Ex: ~~"Select 'myname is' || SMITH from dual;"~~

* Select 'my name is' || ename from emp;

→ Write a Query to display in the below format.

SMITH is earning 800 → (salary)

Ex: Select 'Ename || 'is earning' || SAL from Emp;

→ Write a Query to display in the below format

Smith is [CLERK]

Ex: ~~Select Ename || 'is' || [Job] from Emp;~~

Select EName || ' is [' || job || ']' from Emp;

→ Write a Query to display in the below format

SMITH is earning 800 and joined on 17-Dec-80

* ~~Select~~ EName || 'is earning' || SAL, ~~Hire date~~ from EMP;

* Select ename || 'is earning' || SAL || 'and joined on' || Hiredate from emp;

→ Write a Query to display in the below format

* SMITH is a [CLERH]
  ALLAN is a [SALESMAN] → * Select EName || 'is a' || Job (without brackets) from emp;

* ~~Select 'My Name is' ||~~

* Select EName || 'is a [' || Job || ']' from emp;

Orderby: By using order by clauss we will be able to rearrange the output in ascending (or) descending order.

* order by asc; * order by desc;

Ex: → Select SAL from emp order by SAL asc;

800
950
1100

→ Select SAL from emp order by SAL desc;

1100
950

→ By default order by is in ascending order.

Ex: Select SAL from emp order by SAL;

o/p: 800
950
1100

Ex: Select Hiredate from emp order by Hiredate;

o/p: 17-Dec-80
20-Feb-81
22-Feb-81

→ Select * from emp order by SAL; ⇒ By executing this Records will be changed. It will execute Total Record not Salary Column.

→ Write a Query to display all the employee details in ascending order of dept no & descending order of Salary.

* Select * from emp order by deptno, ~~order by~~ SAL order by desc;

Note: we cannot write order by in multiple times.

* Select * from emp order by deptno, SAL desc;
↳ It will execute only if there is a possibility.

→ Write a Query to display 5th Column of employee table in desending order.

* Select * from emp order by 5 desc;

o/p: EMPNO ENAME JOB MGR Hiredate SAL COMM DEPTNO
23-May-87
23-Jan-82

→ Select hiredate from emp order by 5; ⇒ error because we already know the Column name.

* Select → It chooses which Column (or) Columns needs to be displayed in the output.

* from → It chooses the Table to fetch the data.

* where → As per the condition provided it executes for each & every row and decides which row to pick & which row to reject from the Table.

Std

SID	SNAME	COURSE
1	RAJA	JAVA
2	RANI	MAVA
3	Vijay	BAVA
4	Neel	BAVA

→ Select * from Std where Sname = "Neel";

→ Select * from Std where Course = "BAVA";

→ Write a Query to display BLAKE Record.

* Select * from emp where EName = 'BLAKE';

→ Write a Query to display all the employee names designation & there Salary for all the clerks.

* Select EName, Job, SAL from emp
where JOB = "CLERK";

→ Write a Query to display all the employee detail who joined on 23-may-87

* Select * from emp where Hire date = '23-may-87';

Relational Operators :-

> < >= <= != (or) <> =

* Relational operators can be used on 'where' condition but not in Select Statement. If we use relational operator in Select Statement then we are going to get an error.

→ Query to display all the employee details who is earning Salary more than 2000.

~~* Select * from emp.~~

* Select Ename from emp where SAL > 2000; → for employee names.

* Select * from emp where SAL > 2000; → for all employee details.

→ Query to display all the employee details who is earning 1500 & less than that.

* Select * from emp where SAL <= 1500;

→ Query to display all the emp details apart from Salesman.

* Select * from emp where Job = 'SALESMAN';

→ Query to display all the emp details who belongs to 30 dept

Select * from emp where DEPTNO = 30;

Logical Operators:

AND → whenever we use 'AND' operator rows get selected if both the conditions are satisfied.

OR → whenever we use 'OR' operator rows get selected if any one condition is satisfied.

NOT → It is an opposite operator

→ Query to display of clerk who belongs to 20th department.

Select * from emp where Job = 'CLERK' AND DEPTNO = 20;

→ Query to display all the details of Salesman who earns SAL more than 1500.

Select * from emp where Job = 'SALESMAN' AND SAL > 1500;

→ Query to display all the emp details who belong to 10 & 30 dept

Select * from emp where DeptNo = 10 ~~And~~ OR DEPNO= 30; => for more than one AND operator cannot be used

→ Query to display all the emp details of SALESMAN & MANAGER.

Select * from emp where Job = 'SALESMAN' OR 'MANAGER'; Job=

→ Query to display all the emp details whose MGRNO is 7698 & 7839.

Select * from emp where MGR = 7698 OR ~~7839~~ MGR = 7839;

Special Operators:

IN: In this the record should be in the same list or same column of whenever we use multiple 'OR' operator then we are going to replacing with the special operator called 'IN'

Ex: Select * from emp where MGR IN (7698, 7839, 766, 7782);

→ Query to display all the emp details of CLERK who belong to 10 & 30 dept.

Select * from emp where Job = 'CLERK' And DeptNo in (10,30);

→ Query to display all the emp details of SALESMAN & MGR who belongs to 30th dept.

① Select * from emp where Job = 'SALESMAN' OR 'MANAGER' In, deptno = 30";

② Select * from emp Job in ('SALESMAN', 'MANAGER') and ~~Job~~ in deptno = 30;

→ Query to display all the emp details who joined in the year 81

① Select * from emp where HIREdate = 81;

② Select * from emp where HIREDATE in (81);

③ Select * from emp where hiredate >= '01-JAN-81' and hiredate <= '31-DEC-81';

BETWEEN:- Whenever the records are present in a range then we use a special operator called "Between".

Ex:- Select * from emp where hiredate between '01-JAN-81' and '31-DEC-81';

→ Query to display all the emp details who is earning Salary 2000 to 3000

① Select * from emp where SAL between "2000" and "3000";

Like:- Pattern Matching

- s% => It derives 0 to n Values
- It derives a single char

Std

SID	Sname	Course
1	Raja	Biology
2	Rani	Zoology
3	Chandra	BOTONY
4	chandi	SOCIAL

Select * from Std
where Sname like 'R%';
Course like '%4';
Course like 'B%4';
Course like '%O%';
Sname like '___';
Course like '_O%';
Sname like 'R_%';

→ Query to display all the emp details whose name starts with S

* Select * from emp where ename like 'S%';

→ Query to display all the emp details whose designation ends with HAN.

* Select * from emp where Job like '%HAN';

→ Query to display all the emp details whose Salary contains 2nd last letter '5'.

* Select * from emp where SAL like '%5_';

→ Query to display all the emp details who joined in the year 81.

* Select * from emp where Hiredate like '%81';

→ Query to display all the emp details who joined in the month of December.

DD-MON-YY

* Select * from emp where Hiredate like '%DEC%';

→ Query to display all the emp details designation contains HAN

* Select * from emp where Job like '%MAN%';

NULL:- * NULL is an empty Space (or) Blank Space.

* NULL means nothing
* NULL will not occupy any Space in the Memory.
* NULL is not equals to zero.
* To achieve NULL you have to use a special operator called 'is'.

Ex:- Select * from emp where Comm is NULL;

→ Query to display all the emp details who does not have Manager number.

* Select * from emp where MGR is NULL;

LO
AND
OR
NOT

SO
In
between
like
is

→ Query to display all the emp details apart from 10 & 30 dept

① Select * from emp where dept no NOT in 10 OR 30;

② Select * from emp where DEPTNO not in (10,30);

→ Query to display all the emp details who is earning SAL less than 1000 or more than 2000.

③ Select * from emp where SAL not between ~~not in~~ 1000 and 2000;

→ Query to display all the emp details apart from the name starts with 'S'.

④ Select * from emp where Ename not like 'S%';

→ Query to display all the emp details who is having COMM.

⑤ Select * from emp where COMM is not null;

Functions:-

① Single Row functions → 6 inputs => 6 outputs
② Multi Row functions / aggregate functions
③ Date Functions

→ Single Row funcs: It will take multiple inputs & gives you corresponding o/p for each i/p.

Ex:- Upper() Lower() InitCap() Concat() Replace()
length() Substr()

* Select Upper('Vijay') from dual;

o/p: UPP

VIJAY

* Select lower('VIJAY') from dual;

o/p: Low
---
vijay

* Select Upper('Vijay'), lower('VIJAY'), initcap('vijay') from dual;

o/p: UPP Low Ini
--- --- ---
VIJAY vijay Vijay

→ Query to display all the enames in lower case & designation details in initial Capital.

(*) Select * from emp where Upper('Ename') and initcap ('Job'); ✓

(*) Select * from emp where Upper('Ename'), initcap('Job'); ✗

(*) Select Upper("Ename"), initcap("Job") from emp; ✓

(*) Select lower("Ename"), initcap("Job") from emp; ✓

Concat(): It is similar to concatenation where we are going to join or merge two or more column values or literals.

Syntax: Select Concat (arg1, arg2)
from Tablename;

arg1 } Colnames/literals
arg2 }

Ex: Select Concat (Ename, Job) from emp;
o/p: SMITHCLERK

→ Query to display in the below format by using concatination.
My Name is SMITH
My Name is JONE

(*) Select Concat ('My Name is', Ename) from emp;

Note: Concat function will take max of 2 arguments.

→ Query to display in the below format by using Concat() funcs
SMITH is earning 800

(*) Select Concat (Ename, 'is earning', SAL) from emp; *

(*) Select Ename, Concat ('is earning', SAL) from emp; ✓

(*) Select Concat (Concat (ename, 'is earning'), SAL) from emp;
1 2
o/p: SMITH is earning 800

→ Query to display in the below format by using Concat function.
SMITH is earning 800 & Joined on 17-DEC-80

(*) Select Concat (Concat (Ename, 'is earning'), SAL) and hiredate 17-Dec

④ Select (Concat (Concat (Concat (ename, 'is earning'), SAL), 'and joined on'), hiredate) from emp;

Concat Concat

Replace():

Replace function will replace or will change old text data to new text data. Your old text data can be column name/ literals.

Syntax:-

Select Replace (aug1, aug2, aug3) from tablename;

aug1 → Column name/ literal

aug2 → which char/ chars to pick

aug3 → Replacement char/ chars

Ex:- Select Replace ('Java', 'J', 'M') from dual;

o/p: Mava

→ Select Replace ('test', 't', 'J') from dual;

o/p: JesJ

→ Select Replace ('testEngineer', 'test', 'web') from dual;

o/p: web engineer

→ Select Replace ('test', 'e', ' ') from dual;

o/p: t st

Note:- Replace function will take max of 3 aug & min of 2 aug: if 3rd aug is not given then it will remove the chars which we have given in the 2nd aug.

→ Select Replace ('test', 'e') from dual;

o/p:- tst

→ Select Replace ('test', 't') from dual;

o/p: es

-> Select replace ('Java', 'J', 'm') from dual;

o/p: Java

-> Query to display all the emp details whose designation contain man without using like operator.

(*) ~~Select * from emp where job = 'man' from emp;~~ x

(*) Select * from emp
where Job != replace (Job, 'MAN');

-> Query to display all the emp details whose name contains 'E' without using like operator.

(*) Select * from emp
where ename != replace (ename, 'E');

Length(): It returns the no of chars present in the given string. Your string can be Colname/literal.

Syntax:-
Select length (Colname/literal) from table name;

Ex: Select length ('VIJAY') from dual;

o/p: 5

-> Query to display no. of chars present in each emp name

(*) ~~Select * from emp where length (ename);~~ x

(*) Select length (ename) from emp;

(*) Select ename, length(ename) from emp;

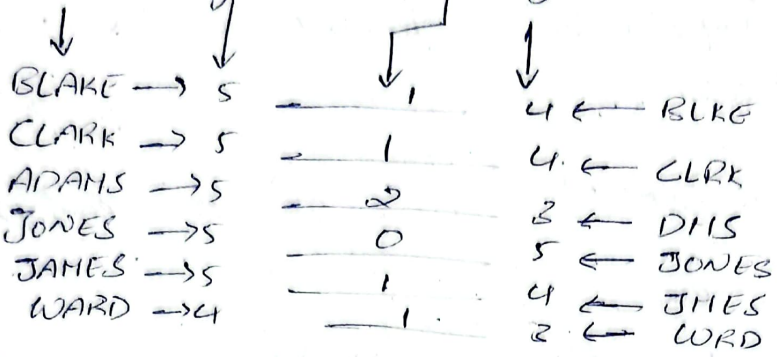
-> Query to display all the emp details where name contains only 5 chars without using like operator.

(*) Select * from emp
where ename != replace(ename, '_____');

(*) Select * from emp
where length(ename) = 5;

→ Query to display no. of a's present in each emp name

* Select ename, length(ename) - length(replace(ename,'A')) from emp;



→ Query to display no. of 'L's present in each ename.

* Select ename, length(ename) - length(replace(ename,'L')) from emp;

Substr():-

It returns the path of characters which is present in the ~~char~~ given String.

Your String can be Colname / literals.

Syntax:- Select Substr(aug1, aug2, aug3)
from tableName;

aug1 → Colname / literal
aug2 → Starting position of a char
aug3 → no. of chars to pick

Q:- Select Substr('Vijay Durga', 7, 5) from dual;

o/p: Durga

→ Select Substr('web engineer', 5, 1) from dual;

o/p:- e

NOTE: Substring will take max of 3rd arg & min of 2nd arg. If 3rd arg is not given then it will execute till the end of the String.

→ Select Substr('web engineer') from dual;

o/p:- Engineer.

-> Select Substr ('web engineer', -5, 2) from dual;

-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
web engineer
1 2 3 4 5 6 7 8 9 10 11 12

o/p: in

-> Select Substr ('web engineer', -5, 2) from dual;

o/p: NULL

-> Query to display 1st char. of emp names

(*) Select Substr (Ename, 1, 1) from emp;

o/p: S

(*) Select Ename, Substr (Ename, 1, 1) from emp;

o/p SMITH S

-> Query to display 1st char & last char of emp Name.

(*) Select Ename Substr (Ename, 1, -1) from emp;

-> Query to display all the emp details whose Designation ends with GER. without using like operator.

(*) Select * from emp where Job = replace (job, 'GER');

(*) Select * from emp
where Substr (Job, -3) = 'GER';

-> Query to display all the emp details whose 2nd last letter of emp name is similary to middle letter of Month.

(*) Select * from emp
where Substr (Ename, -2, 1) = Substr (hiredate, 5, 1);

o/p:- TURNER 08-SEP-81
JAMES 03-DEC-81

Multi Row functions / aggregate functions:

It will take multiple o/p's & gives a single o/p.

→ Max( )
→ Min( )
→ Sum( )
→ Avg( )
→ Count( )

→ Select min(SAL) from emp;
o/p: 800

→ Select min(Ename) from emp;
o/p: Adams

→ Select max(hiredate) from emp;
o/p: 23-May-87

→ Select avg(SAL) from emp;
o/p: 2073.21429

→ Select Sum(Ename) from emp;
o/p: Error

	Numbers	Strings	Date
Max	✓	✓	✓
Min	✓	✓	✓
Sum	✓	x	x
avg	✓	x	x
Count	✓	✓	✓

→ Select Count(COMM) from emp; ← for this it will not execute NULL values.
o/p: 4

→ Select Count(*) from emp;
o/p: 14

→ Select Min(SAL), Max(SAL) from emp;
o/p: 800 5000

NOTE: We can have a combination of single Row function & a Colname in Select Statement. But we cannot having combination of Multi Row function & a Colname in Select Statement.

If we write Multi Row function and a Colname in Select Statement then we are going to get an "error"

→ Select ename, min(SAL) from emp;

o/p: Error → not a Single-group group function.

→ Query to display avg Salary among Clerk

→ Query to display total Salary of 20 Dept

(*) Select ~~Avg~~ Sum (sal) where Deptno = 20 from emp;

(*) Select Sum (sal) from emp where Deptno = 20;

→ Query to display avg Salary of Salesmans.

(*) Selet avg (sal) from emp where Job = "SALESMAN";

→ Query to display no of emp who is earning COMM.

~~(*) Select Count (COMM) from where emp where x~~

(*) Select Count (ename) from emp where COMM is NOT NULL;

(*) Select Count (comm) from emp;

→ Query to display display the date who joined 1st in the year 81.

(*) Select ename, min (hiredate) from emp where hiredate like '%81';

~~(*) Select * from emp where hiredate like ('%81';)~~

o/p: 20-FEB-81

DISTINCT: It is used to get unique values which is present in a column.

Ex: Select distinct (Job) from emp;

o/p:- CLERK
SALESMAN
PRESIDENT
MANAGER
ANALYST

(**) distinct (1st) → Column Name (2nd) in distinct keyword

Ex: Select ename, distinct (deptno) from emp; *

NOTE: We can have a Combination of distinct Colname & Colname in Select Statement.

Eg: Select distinct (Job), ename from emp;

→ Query to display unique job & unique deptno from emp table

(*) Select distinct (Job), deptno from emp; ✗

(*) Select distinct (Job, deptno) from emp; ✗

(*) Select distinct (Job), distinct (deptno) from emp; ✗

Note:- We cannot have more than one distinct function in Select Statement and also we cannot have more than one column in distinct function.

→ Query to display least salary of clerk who joined in the month of december.

(*) Select min (Sal) from emp where Job = 'CLERK' and hiredate like '%DEC%';

→ Query to display no. of employees present in each dept.

(*) Select Count (*(or) Ename) from emp group by deptno;

(*) Select distinct (Ename), deptno from emp;

Groupby:-

(*) Groupby clause divides the table into multiple groups.

(*) Groupby clause will give one op for each group.

(*) We can have a combination of multirow function & Colname in Select Statement only when if we make use that Column in group by clause.

Ex:- Select Count (*), deptno
from emp
Group by deptno;

Op:-

Count (*)	deptno
6	30
5	20
3	10

Ex:- Select Count (*), ename from emp group by deptno;

error:- Not a group By expression.

→ Select ename from emp group by deptno;

o/p: error

→ Select deptno from emp group by deptno; [* → emp]

o/p: 30
20
10

→ Select ename, deptno from emp group by deptno;

o/p: error

→ Select * from emp group by deptno;

o/p: error.

→ Query to display total Salary of each job.

* Select Sum (Sal), job from emp group by job;

o/p:-

Sum(SAL)	Column
4150	CLERK
5600	SALESMAN
8275	MANAGER

→ Query to display highest Sal of each job except Clerk.

* Select Max (Sal), Job from emp where job != 'CLERK' Group by job;

o/p:-
1600 SALESMAN
5000 PRESIDENT.

→ Query to display no. of employees present in each department only if their names contains letter 'A'.

* Select Count (*), deptno where ename like '%A%' group by deptno;

o/p:-

Count(*)	deptno
5	30
1	20
1	10

→ Query to display avg Salary of each job except for those whose designation Contains man.

* Select avg (Sal), Job from emp where Job not like '%MAN%' Group by Job;

→ ~~Query to display~~

NOTE:- * We can write Single row functions in Select Statements as well as in Where Clause

* Multirow functions Can be used only in Select Statements but not in Where Clause.

* If we use multirow functions in Where Clause then we are going to get an error.

Having :-

* Whenever there is a condition for improve then we have to use having clause
* Having clause work with only with groupby.
* Having clause produces each o/p for each group.
* In having clause we can use multirow function.

Ex :- (WAQTD) No. of emps present in each department. Display only those deptno's who have minimum 4 emps working in that deptno?

```
Select Count(*), deptno
    from emp.
Groupby deptno
having Count(*)>=4;
```

O/p :-

Count(*)	Deptno
6	30
5	20

→ Query to display avg. Salary of each job and that avg Salary should be more than 2000

```
Select avg(Sal), job
    from emp
Groupby job
having avg(Sal)>=2000;
```

→ Query to display whose names are repeated twice

```
Select ename from emp
    Group by ename
having Count(ename) = 2;
```

Format of SQL Query

Select
from
where
Groupby
having
Orderby

Execution flow of SQL Query

Select → (5)
from → (1)
where → (2)
Groupby → (3)
having → (4)
Orderby → (6)

Ex:- Select ename, job, sal as Salary from emp
where job = 'SALESMAN' and Salary > 1000;

O/P:- error

→ Select ename, job, sal as Salary from emp order by Salary;

O/P:-

Ename	Job	Salary
SMITH	CLERK	800

→ Query to display total salary of each except clerk that total should exceed 4000 & the o/p should display in descending order of a total Salary.

(*) Select Sum(sal), job from emp
where job != 'CLERK'
group by job
having Sum(Sal) > 4000
order by Sum(Sal) desc;

O/P:-

Sum(SAL)	JOB
8275	MANAGER
6000	ANALYST
5600	SALESMAN
5000	PRESIDENT

→ QTD all employee details whose names starts with Vowels.

(*) Select * from emp
where Substr(ename, 1, 1) in ('A', 'E', 'I', 'O', 'U');

O/P:-

* all columns
ALLEN
ADAMS

→ QTD all employee details whose names starts with ~~Consonants~~ Consonants

(*) Select * from emp
where Substr(ename, 1, 1) not in ('A', 'E', 'I', 'O', 'U');

O/P:- SMITH
KING

(**) QTD whose name starts with Vowels and ends with Consonants.

(*) Select * from emp where Substr(ename, 1, 1) in ('A', 'E', 'I', 'O', 'U') and
Substr(ename, -1, 1) not in ('A', 'E', 'I', 'O', 'U');

O/P:- ALLEN
ADAMS

→ QTD whose name starts with Consonants and ends

* Select * from emp
  where Substr(Ename,1,1) notin ('A','E','I','O','U') and
  Substr(Ename,-1,1) in ('A','E','I','O','U'); O/P: BLAKE

Sub-Query:- Whenever there is an indirect condition then generally we go for Sub Query. Whenever we write Sub-Query Atleast one Column data should exist between the tables.

There are two types of Sub-Queries:

→ Single Row Sub-Query.

→ Multi Row Sub-Query.

* Single-Row Sub Query:- Single Row Sub-Query will ~~Show~~ return a single output. Hence we use Relational Operator (>, <, >=, <=, =, !=)

* Multi-Row Sub Query:- It will Return multiple output. Therefore we use "in".

Faculty

SNO	ENAME	COURSE	SAL
1.	SUDIP	JAVA	100
2.	DEEPAK	JAVA	200
3.	POOJA	SQL	200
4.	SALMAN	Social	100
5.	JEEVAN	SQL	200

* Select * from faculty
  where SAL = (Select Sal from faculty
  where Ename = "POOJA");

[QTD whose SAL is Similar to POOJA]

→ QTD whose Course is Similar to Sudip

* Select * from faculty
  where Course = (Select Course from faculty
  where ename = 'SUDIP');

→ QTD all employee details whose DEPTNO is Similar to ALLEN

* Select * from faculty
  where DEPTNO = (Select Deptno from emp where ename = "ALLEN");

O/P:- ALLEN 30
WARD 30
MARTIN 30

→ QTD all the employee details whose designation is similar to the employee whose empno = 7900

① Select * from emp
where job = (Select job from emp where empno = 7900);

O/P:- EMPNO JOB
7369 CLERK
7900 CLERK

→ QTD all the employee details whose salary is more than average salary of 10th Department.

② Select * from emp
where sal > (Select avg(sal) from emp
where deptno = 10);

O/P:- SAL
2975
3000
5000
3000

→ QTD all employee details whose salary is more than MARTIN SALARY and designation is similar to table

③ Select * from emp
where sal > (Select sal from emp where ename = 'MARTIN')
and job = (Select job from emp where ename = 'BLAKE');

O/P:- ENAME JOB SAL
JONES MANAGER 2975
BLAKE MANAGER 2850
CLARK MANAGER 2450

Eg:- Select * from emp where SAL > 1500 and job = 'MANAGER';

→ QTD all the employee details who joined before TURNER

④ Select * from emp
where hiredate < (Select hiredate from emp
where ename = 'TURNER');

O/P:- HIREDATE
17-DEC-80
20-FEB-81

* Between Two Tables based on Common Column :-

→ QTD work location of 'TURNER'

* Select Loc from dept
  where deptno = (Select deptno from emp
  Where ename = 'TURNER');

O/P :- Loc : CHICAGO

→ QTD all the employee details who belongs to accounting department

* Select * from emp
  where deptno = (Select deptno from dept
  where Dname = 'ACCounting');

→ QTD all emp details who belongs to newyork and Dallas

* Select * from emp
  where deptno in (Select deptno from dept
  where loc in ('NEWYORK', 'DALLAS'));
  Multirow Sub-Query

→ QTD all employee details whose location Contains letter 'O' in it

* Select * from emp
  where deptno in (Select deptnofrom dept
  where loc like '%O%');

→ QTD all the emp details of Sales and Research department

* Select * from emp
  where deptno in (Select deptno from dept
  where Dname = 'SALES' or Dname = 'RESEARCH');

→ QTD all the employee details whose Salary is less than miller.

* Select * from emp
  where SAL < (Select Sal from emp where ename = 'MILLER');

→ QTD all employee details who belongs to 10th & 30th dept

* Select * from emp
  where deptno in (10, 30);

emp

empno	ename	MGR
1	P	4
2	Q	1
3	R	2
4	S	3

→ Select * from emp
where empno = (Select MGR from emp
where ename = 'P');

→ Select * from emp
where MGR = (Select empno from emp
where ename = 'R');

→ QTD Managers details of 'SCOTT'

(*) Select * from emp
where empno = (Select MGR from emp
where ename = 'SCOTT');

→ QTD least Salary employee details

(*) Select * from emp
where Sal = (Select min(Sal) from emp);

→ QTD ALLEN's team leads, n details.
(team leads)

(*) Select * from emp
where empno = (Select MGR from emp where empno = (Select MGR
from emp where ename = 'ALLEN'));

→ QTD all the team lead details who have minimum 3 employees
working under them.

(*) Select * from emp
where empno in (Select MGR from emp group by MGR
having Count(MGR) >= 3);

→ Query to display all the family details of James and Jones.

(Mgr)

* Select * from emp where empno in (Select MGR from emp where ename in ('JAMES', 'JONES'));

Joins:-

* Joining or Merging one (or) two tables is called "Joins".
* Joins are used to fetch the data from multiple Tables.

Types of Joins:-

→ Cross Join / Cartesian Join:-

In this Join we can able to join one (or) two tables

In this type of Join each & every record of one table is going to match with each & every record of another table.

Ex:- If table A contains six records & table B contains 4 records then the o/p will be 24 records.

emp	dept
deptno =	deptno
10	10
20	20
30	30
40	50
4 rows	4 rows

* Select * from emp, dept;

o/p: 4×4 = 16

NOTE:- Whenever we use Cross Joins we always gets Correct data as well as Incorrect data. So in Real time Scenario we will never use "Cross Joins".

→ Inner / Equi Join:- Whenever we use equi join we always gets matched records.

* With the help of equi joins we can able to fetch the data from two tables.
* In equi join there should be a Common Column exists b/w the table and we have to provide a Condition.

Emp	dept
deptno	deptno
10	10
20	20
30	30
40	50
4rows	4rows

=> Select * from emp, dept
Where emp.deptno = dept.deptno;

o/p:- 10 - 10
20 - 20
30 - 30

→ Query to Display all the Enames & their work location

~~Select Ename, loc where~~

~~Select * from emp, dept where Ename~~ emp.ename = dept.loc;

Select * from emp, dept where emp.deptno = dept.deptno;

Select ename, loc from emp, dept
Where emp.deptno = dept.deptno;

o/p:-

Ename	Loc
Smith	DALLAS

→ Query to display emp details & dept details of TURNER

Select * from emp, dept
Where emp.deptno = dept.deptno and ename = 'TURNER';

→ Query to display dept details of all the CLERKS.

Select * from emp, dept
Where emp.deptno = dept.deptno and Job = 'CLERK';

→ Query to display emp information & dept information for all the employee who is having adams designation.

Select * from emp, dept
Where emp.deptno = dept.deptno
And
Job = (Select Job from emp where ename = 'ADAMS');

→ Query to display only emp details who belongs to accounting dept.

① Select * from dept where Dname = 'ACCOUNTING';

② Select emp.* from emp, dept
where emp.deptno = dept.deptno
and
Dname = 'ACCOUNTING';

③ Select * from emp where deptno = (Select deptno from dept
where Dname = 'ACCOUNTING');

→ Query to display only department information of JONES

① Select dept.* from emp, dept
where emp.deptno = dept.deptno
and
Ename = 'JONES';

→ Query to display only emp information who belongs to 20th dept.

① Select emp.* from emp, dept
where emp.deptno = dept.deptno
and
deptno = 20;

② Select * from emp where deptno = 20;

Outer Joins:

Whenever we use equi joins we always gets matched records to get matched records & un-matched records we always go for outer joins.

There are 3 Types of Outer Joins:-

① Left OJ
② Right OJ
③ Full OJ

→ In left outer join all the records of left side table & matched records of right side table will display.

emp	dept
deptno	deptno
10	10
20	20
30	30
40	50

Eg:- Select * from emp, dept
where emp.deptno = dept.deptno(+);

O/P:- 10 - 10
20 - 20
30 - 30
40 - —

→ In Right outer join all the records of right side table and matched records of left side table will display.

Eg:- Select * from emp, dept
where emp.deptno(+) = dept.deptno;

O/P:- 10 - 10
20 - 20
30 - 30
— - 50

→ In full outer join all matched & unmatched records of both the tables will display.

Eg:- Select * from emp, dept
where emp.deptno(+) = dept.deptno(+);

O/P:- Error

*NOTE:- full outer joins cannot be achieved as per oracle standards. Hence we use ANSI (American National Standard Institute) Rule.

ANSI (American National Standard Institute):-

Cross Join:-

```
Select *
  from emp Cross Join dept;
```

Inner Join:-

```
Select * from emp Join dept
on emp.deptno = dept.deptno;
```

L.O.J Join:-

```
Select * from emp L.O.J dept
on emp.deptno = dept.deptno;
```

R.O.J Join:-

```
Select * from emp R.O.J dept
on emp.deptno = dept.deptno;
```

F.O.J Join:-

```
Select * from emp F.O.J dept
on emp.deptno = dept.deptno;
```

Self Join:-

* Joining the table to itself is called "Self Join".
* Without aliasing we cannot achieve "Self Join".

Eg:-
```
Select *
from emp e, emp i
where e.MGR = i.empno;
```

emp e

empno	ename	MGR
1	A	4
2	B	1
3	C	2
4	D	3

emp i

empno	ename	MGR
1	A	4
2	B	1
3	C	2
4	D	3

lead

→ To find emp information & teamlead information of Jones

* Select * from empfoin (Select MGR from emp
where ename in ('Jones'));

* Select * from emp e, emp i
where e.MGR = i.empno
and
e.ename = 'JONES';

O/P:-

MGR	ename	empno
7839	JONES	7566

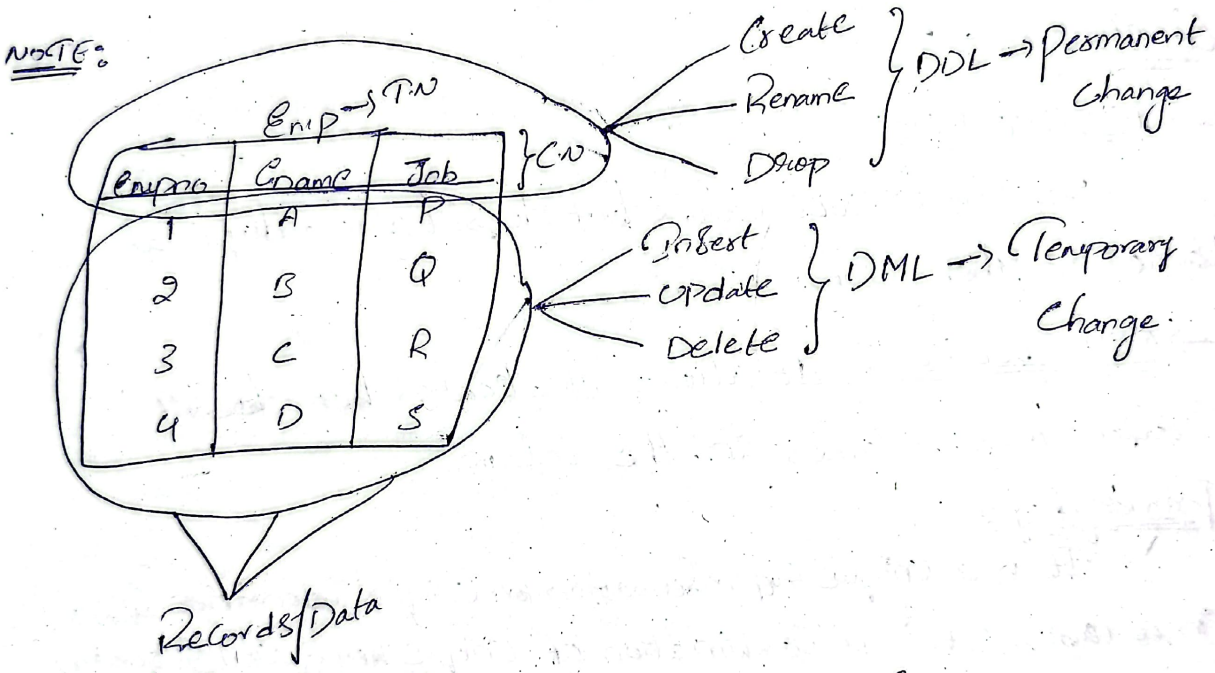
→ Query to display who earns as who gets same salary.

```
Select * from emp e, emp i
    where e.Sal = i.Sal
        and
    e.ename != i.ename;
```

→ Query to display who joined in the same date.

```
Select * from emp e, emp i
    where e.hiredate = i.hiredate
        and
    e.ename = i.ename;
```

```
Select e.hiredate, e.ename from emp e, emp i
    where e.hiredate = i.hiredate
        and
    e.ename != i.ename;
```



DDL (Data Defination Language) :- By using DDL Command we will be able to Create a table, Modifie a table, Rename a table, Drop the table information from the Data Base.

All DDL Commands are permanent change. It is used to define the data base & its Structure.

To Create a Table we should know two things. They are

→ Data Types

→ Constraints.

Data Types:-

→ Number( ):-

Whenever we are Setting the data types as Numbers then it is always Numerical in Nature.

→ CHAR( ):-

Whenever we are Setting the data types as characters it is always alphabetical in Nature.

→ VARCHAR( ):-

It is present in the previous Version of data Base & for the latest Version we will use "VARCHAR2( )" & it is always alphanumerical in Nature.

Constraints :-

→ UNIQUE KEY:-

It allows NULL Values but doesn't allow duplicate values in the Column.

→ NOT NULL KEY:- It allows duplicates but doesn't allow NULL Values in the Column.

Primary key:-

It is a unique representation (or) unique identifier Column in a table. It is a Combination of unique key & NOT NULL key. Which blocks unique Values & NULL Values.

In a table there Should be one primary key Column

Difference between primary key and unique key.

Primary	~~foreign~~ unique key
→ In a table we can have only one primary key column	→ In a table we can have multiple ~~foreign~~ unique key Column.
→ It blocks duplicate & Null Values	→ It Blocks duplicate & allows Null Values.

foreign key:-

→ It is also known as Referential Integrity Constraints.

→ The keyword which we use for foreign key is "References".
→ Foreign key builds a relationship between two Tables.
→ Foreign key in the current table is called as child table & the referencing table we call it is as parent table (or) master table.

Candidate key:
→ The colums which are eligible to become primary key is called Candidate key.

Alternate key:
→ After choosing a column as primary key then the remaining Candidate key we call it as "alternate key".

Syntax for Creating a Table:
Create Table Name
(Colname datatype(Size) Constraint,
" " "
" " ");

Eg:- Create table hebbal
(uno number(5) unique,
uname char(15) not null);

O/P:- Table_Name Table_Type
HEBBAL TABLE

Syntax for Rename a Table:
Rename old Tablename to new TableName;

Eg:- Rename hebbal to nagavara;
O/P:- Table renamed.

Syntax for Drop a Table:
Drop Table Table-name;

Eg:- Drop table Nagavara;
O/P:- Table Dropped.

ALTER:

ALTER Command is used to do modification to an existing table. It might be adding a column/renaming a column (or) dropping a column to an existing table.

Syntax for how to add a Column:

ALTER Table Tablename
add(Colname datatype(Size) Constraint);

Eg: alter table nagavara
add(mobno numbers(10) unique);

O/P: Table altered

→ desc nagavara

Name
UNO
Uname
mobno

Syntax for How to Rename a Column:-

ALTER Table TableName
Rename Column oldColname to newColname;

Eg:- alter Table Nagavara
Rename Column mobno to mno;

O/P:- Table altered

→ desc nagavara

Name
UNO
Uname
HNO

Syntax for how to dropping a Column:-

ALTER Table Tablename
drop Column Colname;

Eg:- alter table nagavara
drop Column mno;

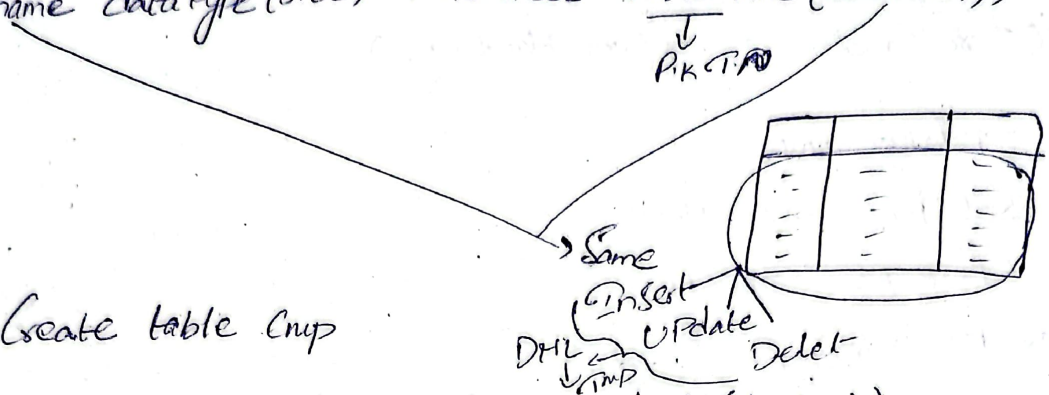
O/P: Table altered

→ desc Nagavara

Name
UNO
UNAME

Syntax for Creating a Table by using foreign key :-

Create Tablename
(Colname datatype (size) references tablename (Colname));



Eg:- Create table Dup
(deptno number (5) reference dept (deptno));

Difference between Drop, Delete & Truncate

→ Drop & Truncate both are DDL Commands and it is a permanent change.

→ Drop will delete the Complete table from the data base whereas Truncate will delete all the records from the Table but table Structure remains Same.

Data Manipulation Language :- (DML)

By using DML Commands we will be able to insert a record to an existing table, updating the record & deleting the record from an existing table.

all DML Commands are temporary change.

Syntax for inserting a Record :-

Insert into TableName
(Col1, Col2, Col3 - - - - -)
Values (V_1, V_2, V_3 - - - - -);

Eg:- Select * from Nagavara =) O/P= UNO UNAME
1 Vijay

Insert into Nagavara
(Uno, Uname)
Values (1, 'Vijay')

Syntax for Updating a Record :-

Update TableName
Set Colname = NewValue
Where Condition; → (Not Mandatory)

Eg:- Update emp
Set job = 'SUPERMAN'
Where job = 'SALESMAN';

Syntax for deleting a Record :-

Delete TableName
where Condition;

Eg: Delete emp
where job = 'SUPERMAN';

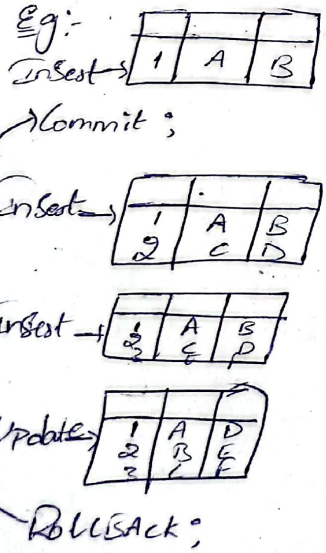
O/P:- 4 rows Deleted.

TCL (Transaction Control Language) :-

Commit :- all DML Commands are temporary Change & to make it as permanent we have to use a TCL Command Called "Commit".

RollBack :- Rollback acts like a Undo function. Whenever we use Rollback it always go to previous Commit.
If Previous Commit is not there it will goto auto Commit.

Eg:-



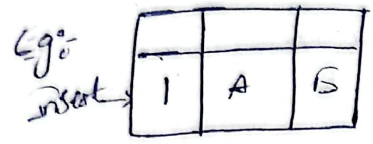
NOTE :- Deleted Values Can be RollBack Whereas Truncated Values Cannot be RollBack.

SavePoint:

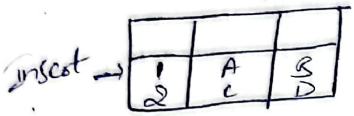
Savepoint acts like a breakpoint b/w Commit Command & RollBack Command.

① Savepoint Savepointname;

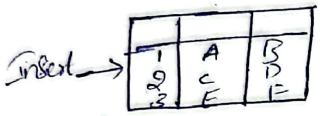
Eg:



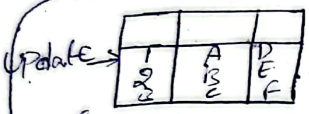
Commit;



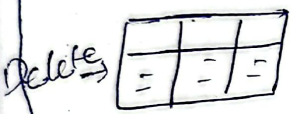
- Savepoint A



→ Savepoint B



- Savepoint C



Rollback to B;

② Show RecycleBin;

FlashBack: Whenever we drop a table it always go to Bin folder.

To Bring back the Table from Bin folder we have to use FlashBack

Syntax:

Flashback Table Tablename to before drop;

Purge: Whenever we ~~want~~ have to delete a Table from bin folder also then we have to use purge Command.

Syntax: Purge Table Tablename;

Eg: Purge Table Nagavarai;

O/P: Table Purged.

DCL (Data Control Language)

→ Grant => Grant Select on emp to hr;

→ Revoke => Revoke Select on emp from hr;

Date functions:-

It is used to change the date format as per the users choice.

To Display the current (Today) we have to use below Query.

Format:

Date	Month	Year
DD	MON	YYYY
	MM	YY

* Select Sysdate
  from dual;

o/p: 15-NOV-23

* Date functions are always temporary change

Eg: Select to_char (hiredate, 'dd/MM/yy') from emp;

o/p: TO_CHAR(
17/12/80
20/02/81

Syntax:

Select To_char (aug1, aug2)
from Tablename;

aug1 → Column/literal
aug2 → User defined format

→ Query to display all the emp joining years in words.

* Select To_char(hiredate, Date/Month/Year) from emp;

* Select To_char(hiredate, 'YEAR') from emp;

o/p: TO_CHAR
NINETEEN EIGHTY

→ Query to display all the emp joining Months in words.

* Selcct To_char (hiredate, 'MONTH') from emp;

→ Query to Display all the emp details who joined in the year 81 by using date functions.

* Selct *from To_char (hiredate, 'YY') where Hiredate like '__-__-81'
from emp;

(*) Select * from emp where hiredate like '---81' To_char (hiredate, 'yy');

~~(*) Selec~~

(*) Select * from emp
Where To-char (hiredate, 'yy') = 81';

→ Query to display all the emp details who joined in the ~~month~~ month of December.

(*) Select * from emp
Where To-char (hiredate, 'Mon') = 'DEC';

→ Query to display all the emp names & their no. of years of experience

(*) Select Ename, To_char (Sysdate, 'yyyy') -
To-char (hiredate, 'yyyy') from emp;

o/p:- SMITH 43

→ Query to display all the emp details who have min 5 years of experience

(*) Select * from emp
where To-char (Sysdate, 'yyyy') -
To-char (hiredate, 'yyyy') >= 5;

→ Query to display duplicate Record.

(*) Select empno from emp Group by empno
having Count (empno) >= 2;

→ Query to display delete duplicate Record

(*) Delete emp where empno in (Select empno from emp
Group by empno
having Count (empno) >= 2);

Co-Related Sub-Queries:

It is similar to Sub-Queries it works for both joins & Sub-Queries.

Rownum:

* Rownum is pseudo column.
* Rownum values are Numeric in Nature & Unique in Nature.

Eg:- * Select rownum from dept;

Rownum
1
2
3
4

* Select rownum from (Select ename, Job, SAL from emp Where Job = 'SALESMAN');

O/P:

Rownum
1
2
3
4

NOTE:- By default, rownum value is '1'.

* Select * from emp
  where rownum=1;

O/P:

empno	ename
7369	SMITH

* Select * from emp
  where rownum=3;
  O/P:- no rows selected.
* Selct * from emp
  where rownum>=3;
  O/P:- No rows selected.

NOTE:-

* RowNum works for: less(<) (or) lessthan(or)equalto(<=)

Ex:- Select * from emp
  where rownum<=4;
O/P:- Display 4 rows.

(*) Select * from emp
where rownum >= 3;

O/P:- No rows Selected.

→ Query to display 3rd record of emp Table.

(*) Select * from emp
where rownum <= 3;

(*) Select * from (Select emp.*, rownum as a from emp)
where a = 3;

O/P:

EMPNO	ENAME	JOB
7521	WARD	SALESMAN

→ Query to display 5th record of emp Table.

(*) Select * from (Select emp.*, rownum as b from emp)
where b = 5;

→ Query to display 4th & 6th record of emp table.

(*) Select * from (Select emp.*, rownum as b from emp)
where b in (4, 6);

→ Query to display last record of emp Table

(*) Select * from (Select emp.*, rownum as b from emp)
where
b = (Select Count(*) from emp);

→ Query to display 2nd last record of emp Table.

(*) Select * from (Select emp.*, rownum as b from emp)
where b = (Select Count(*) - 1 from emp);

nth highest Sal

```
Select min(Sal)
from (Select distint(Sal)
      from emp
      Orderby Sal(desc)
Where rownum<=3;
```

Sal
800
300
600
400
100
100
200
500
500

800	300
600	600
500	400
400	100
300	200
200	500
100	

nth least Sal

```
Select max(Sal)
from (Select distint(Sal)
      from emp
      Orderby Sal)
Where rownum<=3;
```

300
600
400
100
200
500

100	100
200	200
300	300
400	400
500	500
600	600

nth highest → Min → desc

nth least → Max → asc