STRUCTURED QUERY LANGUAGE

→ What is sql?

- It is a query language.
- SQL supports the in-built commands to interact with database.
- SQL commands are classified into 5 types.

1. DDL Commands (Data Definition Language)

– The DDL Commands further classified into 5 Commands. They are listed below.

① Create Command.
② Alter command.
③ Rename Command.
④ Truncate command.
⑤ Drop command.

→ Let us know about DDL Commands.

Data Definition Language Commands (DDL):

- The DDL Commands are useful to do the following actions :-

1. To create a new database object like Tables, views, index and sequence etc.
2. And to make changes in the structure of a Data Base.
3. To Drop (DELETE) the database object permently from the system.
4. To change the name of database object.
5. To erase All the records of a Table.

• SQL has 5 DDL commands let us know them detail:

① Create command:-

• The create command is used for Table creation

Syntax:-

```
SQL> CREATE TABLE TABLE_NAME (
        COLUMN_NAME_1  DATA_TYPE (WIDTH),
        COLUMN_NAME_2  DATA_TYPE (WIDTH),
        :
        COLUMN_NAME_N  DATA_TYPE (WIDTH));
```

Example:-

Let us create a student table.

Columns and Data TYPES in Student Table

RNO — means digits so — Number(3)
here 3 means -999 to +999

NAME — String — we prefer varchar2(10)

GENDER — string — we prefer char(1) means only [M/F] if we use char(6) means [MALE/FEMALE]

COURSE — string — varchar2(10)

DOJ — Combination of words + digits — we use [DATE] datatype

FEE — digits — Datatype is number(5)

MOBILE — It is standard 10 digit number — so, number(10)

ADDRESS — combination of words + digits — varchar2(20 (or) 30)
(or)
10

Now, let use create a student table,

```
SQL > CREATE TABLE STUDENT(RNO NUMBER(3),
      NAME VARCHAR(20), GENDER CHAR(1),
      COURSE VARCHAR2(20), DOJ DATE,
      FEE NUMBER(5), MOBILE NUMBER(10),
            ADDRESS VARCHAR(30));
```

Table created.

SQL >

→ Now to see the structure of a table use

desc Student keyword.
↳ description.

Now

```
SQL > DESC STUDENT;
```

Name	Null?	Type
RNO		NUMBER(3)
NAME		VARCHAR(20)
GENDER		CHAR(1)
COURSE		VARCHAR2(20)
DOJ		DATE
FEE		NUMBER(5)
MOBILE		NUMBER(10)
ADDRESS		VARCHAR(30)

SQL>

CREATING NEW TABLE FROM EXISTING TABLES:

SYNTAX:-

```
CREATE TABLE NEW_TABLE_NAME AS SELECT
  * | COLUMNS_LIST FROM OLD_TABLE_NAME
  WHERE [CONDITION];
```

Example:-

```
SQL> CREATE TABLE STD01 AS SELECT * FROM
                        STUDENT;
```

Table created.

→ we can select a particular columns from the existing

Example:-

```
SQL> CREATE TABLE STD02 AS SELECT
     NAME, COURSE, FEE from STUDENT;
```

```
[SQL> CREATE TABLE STD_COURSE AS SELECT
     * FROM STUDENT WHERE] X
```

```
SQL> CREATE TABLE EMP_CSE AS SELECT
     * FROM EMP WHERE DEPT = 'CSE';
```

(2) Alter COMMAND:-

- The Alter command is used to do the following operations on the table.

1. Adding one or more new columns to the table.
2. Modifying one or more columns datatype and width of a table.
3. To change the name of the columns of a Table.
4. To DROP (DELETE) one or more columns of a Table.
5. To Add constraints to the columns of a Table.
6. To Drop constraints to the columns of a Table.

ADDING COLUMNS:-

- We can add one or more new columns to the existing table.

SYNTAX:-

ALTER TABLE TABLE_NAME ADD
(COLUMN_NAME DATATYPE(WIDTH)...);

Example:-

ALTER TABLE STUDENT ADD (STUDENT_INITIAL
varchar2(10));

MODIFYING COLUMNS:-

- We can change the Datatype AND/OR width of the one or more existing columns.

NOTE:

- If the table is empty then only we can change the Datatype AND/OR we can change the width of the columns.

SYNTAX:- AITER TABLE TABLE_NAME MODIFY
(COLUMN_NAME_1 NEW_DATATYPE(NEW_WIDTH));

Example:-

```
SQL> ALTER TABLE STUDENT MODIFY
        (STUDENT_INITIAL CHAR(1),
         GENDER VARCHAR2(6));
```

Renaming Columns:-

- We can, Also change the name of the columns.
- AT a time, we can change only one column name.

SYNTAX:-

```
ALTER TABLE TABLE_NAME RENAME COLUMN
    OLD_COLUMN_NAME TO NEW COLUMN_NAME;
```

Example:-

```
SQL> ALTER TABLE STUDENT RENAME COLUMN
        FEE TO COURSE_FEE;
```

DROPING COLUMNS:-

- We can drop(DELETE) one or more columns of a Table.
- AT a time, we can drop only one column.

SYNTAX:-

```
DROP ALTER TABLE TABLE_NAME DROP COLUMN
                    COLUMN_NAME;
```

Example:-

```
SQL> ALTER TABLE STUDENT DROP COLUMN
                    STUDENT_INITIAL;
```

SQL> ALTER TABLE STUDENT DROP COLUMN ADDRESS;
SQL> ALTER TABLE STUDENT DROP COLUMN GENDER;

③ RENAME COMMAND:-
- The Rename command is used to change the name of a Database object.

SYNTAX:-
RENAME OLD_NAME TO NEW_NAME;

Example:-
SQL> RENAME STUDENT TO STUD;

④ TRUNCATE COMMAND:-
- The Truncate command is used to erase all the records of a table permanently.
- once the table was truncated, then we can't get back the Records.

SYNTAX:-
TRUNCATE TABLE TABLE_NAME;

Example:-
SQL> TRUNCATE TABLE STUDENT;

⑤ DROP COMMAND:-
- To DROP [DELET permanently] Any Database object permanently from the system use this drop command.

SYNTAX:-
DROP TABLE TABLE_NAME;

Example:-
SQL> DROP TABLE STUDENT;

Ex: DROP TABLE x;

DML Commands :-

. The DML commands are used for the following Transaction.

1. Inserting Records into view/Table.
2. Updating existing records with new values.
3. Deleting unwanted records from Table/views.

. SQL Has 3 DML commands such as.

1. Insert (Inserting Records)
2. Update (Updating Records)
3. Delete (Deleting Records).

Insert Command:-

Form1:

. use this form to insert values into specific columns of a Table/view

SYNTAX:

```
INSERT INTO TABLE_NAME(COLUMNS_LIST)
        VALUES (VALUES_LIST);
```

Ex:

```
INSERT INTO MPC(NAME, M1, M2, M3)
        VALUES ('RAVI', 70, 80, 90);
```

or

```
insert into MPC values
('&name', &m1, &m2, &m3);
```

/

Form 2:

- use this form to insert values into All the columns of a Table / view

SYNTAX:

INSERT INTO TABLE_NAME VALUES(VALUES_LIST);

Ex:-

INSERT INTO VOTER (101, 'RAVI', 34);

FORM 3:

- use this form for Data entry Purpose.
- In this form we should provide all the column values

SYNTAX:

INSERT INTO TABLE_NAME VALUES
(& COLUMNS_LIST);

Ex:

INSERT INTO VOTER VALUES (&ID, '&NAME', '&AGE');

NOTE:-

'/' — Execute the Previous command use slash command

Inserting Records from other Table:

SYNTAX:

INSERT INTO TABLE_NAME1 [(COLUMS_LIST)]
SELECT * | COLUMNS_LIST FROM TABLE_NAME2
[WHERE CONDITION]

Ex:

1. INSERT INTO VOTER_LIST(NAME) SELECT NAME FROM VOTER;

2. INSERT INTO VOTER_LIST SELECT * FROM VOTER.

INSERT INTO VOTER_LIST SELECT * FROM VOTER
WHERE NAME LIKE 'R%' OR NAME LIKE 'A%'
ORDER BY NAME;

Update:

- The Update Command is used to update the records of a Table/View.
- with the update command

[insert into student values (&sid, '&sname', &m1, &m2, &m3);]

error

ORA-00947: not enough values.

ORDER BY NAME;

19/9

Update:- Update command is used to update the Records of a table / view.

* With update command, we can do the following changes.
* We can ~~change~~ Replace An EXISTing value with a NEW value of a COLUMN.
* We can calculate A COLUMN value from the value of other columns.

SYNTAX:- Update; Table-Name SET COLUM-Name1=
COLUM-Name-2=VALUE
;
COLOM_Name N=VALUE
WHERE CONDITION],

WITH out WHERE clause, IF WE update A COLUMN
records of the Table will update

WITh wese clause, only The records which Are SATISFYING The

Update CEC SET TOTAL = M1+M2+M3;
Update CEC SET M1 = 77 wese Name = 'Ramu';
Update MEC SET TOTAL = M1+M2+M3, Avarge = (M1+M2+M3)
" " " Result = 'pass' where M1 > 40 And M2
M3 > 40 ;

Update MEC SET Result = 'Fail' when M1 < 40 or M2 < 40 M3 < 40;
Update MEC SET Grade = 'A' where Avarge >= 85;
" " " = 'A' where Avarge < 85 and Avarge
" " = 'B' where Avarge < 60 and avarage >= 50
" = 'C' where Avarge < 50 and avarge >= 40;

Delete:- The Delete command is used to Delete un wanted Records from the Table/View.

* If we Erase the records by using terminate command, we cannot back. The records but IF we Deleted Records using delete command we can ACCESS The Deleted Records.

SYNTAX:- Delete from Table-Name [where cond:

without wese clause, ALL Records of A Table will be deleted.

With where clause, only the records, which are SATISFYING the specified condition will be deleted.

Ex:-

~~Select~~

Delete from CEC where Name = 'Ramu';

Delete from CEC;

DQL Command:-

- The DQL command is used to select/Fetch/Access All or specific records from the table/view.

SYNTAX:

```
SELECT [DISTINCT|UNIQUE] * |COLUMNS_LIST FROM
    TABLE_NAME [WHERE ROWNUM CONDITION]
        [GROUP BY COLUMNS_LIST]
        [HAVING CONDITION]
        [ORDER BY COLUMNS_LIST ASC|DESC];
```

- Select is the DQL command in SQL.
- The UNIQUE/DISTINCT CLAUSE will displays/selects only the unique Records from the specified table.

Ex:

```
SELECT UNIQUE * FROM STUDENT;
SELECT DISTINCT * FROM STUDENT;
SELECT * FROM EMP;
SELECT NAME, DEPT, SALARY FROM EMP;
```

- The ROWNUM CLAUSE DISPLAYS SPECIFIED Number of Records from the Top.

Ex: `SELECT * FROM EMP WHERE ROWNUM <= 3;`

- The 'WHERE' CLAUSE selects only the Records which are satisfying the specified condition/condition

Ex:- `SELECT * FROM EMP WHERE DEPT = "CSE";`

Select * from EMP WHERE DEPT = "CSE' AND GENDER = 'F';

Select * FROM EMP WHERE SALARY BETWEEN 40000 AND 50000;

. The Group by clause groups the similar type records into one group.

Ex:-

SELECT DEPT, MAX(SALARY) FROM EMP GROUP BY DEPT;

SELECT GENDER, MIN(SALARY), MAX(SALARY) FROM EMP WHERE DEPT = 'CSE' GROUP BY GENDER;

. The HAVING CLAUSE Filters the group Records.

Ex:-

SELECT DEPT, MIN(SALARY), MAX(SALARY) FROM EMP GROUP BY DEPT HAVING DEPT = 'CSE' OR DEPT = 'IT';

SELECT DEPT AS "DEPARTMENT", MIN(SALARY) AS "MINIMUM_SALARY", MAX(SALARY) AS "MAXIMUM_SALARY" FROM EMP GROUP BY DEPT HAVING DEPT='CSE' OR DEPT = 'IT';

. The order by clause displays the records in am orde

Ex:-

SELECT * FROM EMP ORDER BY NAME;

SELECT * FROM EMP ORDER BY NAME DESC;

SELECT * FROM EMP ORDER BY NAME ASC;

Operators in SQL:-

- In SQL operators are the special symbols or words which are used to perform a specific operations on values
- SQL has 4 types of operators such as;

1. Arithmetic operators (+, -, *, /)
2. Relational operators (<, <=, >, >=, =, != and)
3. (Between, NOT Between, IN, NOT IN, LIKE, NOT LIKE, ANY, NOT ANY, ALL, NOT ALL, EXISTS, NOT EXISTS)
4. Logical operators (AND, OR, NOT)
5. SET operators (UNION, UNION ALL, INTERSECT, MINUS)

ARITHMETIC operators:-

Ex:-

```
SELECT  45+67+52 FROM DUAL;
SELECT  56-73 FROM DUAL;
SELECT  5*3*5 FROM DUAL;
SELECT  10/3 FROM DUAL;
```

Relational operators:-

```
SELECT * FROM EMP WHERE WORK_EXP<5;
UPDATE EMP SET SALARY= SALARY+5000
        WHERE SALARY<40000;
DELETE FROM EMP WHERE WORK_EXP<3;
```

LOGICAL operators:-

UPDATE MEC SET Result = "PASS" WHERE M1 >= 40 AND M2 >= 40 AND M3 >= 40;

UPDATE MEC SET Result = "FAIL" WHERE M1 < 40 OR M2 < 40 OR M3 < 40;

UPDATE MEC SET M3 = 69 WHERE NAME = 'RAM' AND M3 = 68;

SELECT * FROM MEC WHERE TOTAL = TOTAL >= 180 AND TOTAL <= 200;

SELECT * FROM EMP WHERE NOT DEPT = "CSE";

SELECT * FROM EMP WHERE NOT DEPT = 'CSE' AND NOT DEPT = 'IT';

→ Alternative:

SELECT * FROM MEC WHERE TOTAL BETWEEN 180 AND 200;

SELECT * FROM MEC WHERE TOTAL NOT BETWEEN 180 AND 200;

→ SELECT * FROM EMP WHERE DEPT = 'CSE' OR DEPT = 'IT';

SELECT * FROM EMP WHERE DEPT IN ('CSE', 'IT');

SELECT * FROM EMP WHERE DEPT NOT IN ('CSE', 'IT');

→ SELECT * FROM EMP WHERE DEPT = 'CSE' AND 'DEPT = 'IT'

SELECT * FROM EMP WHERE DEPT = ALL ('CSE', 'IT');

SELECT * FROM EMP WHERE NOT DEPT = ALL ('CSE', 'IT');

→ SELECT * FROM EMP WHERE DEPT = 'CSE' OR DEPT = 'IT';

SELECT * FROM EMP WHERE DEPT = ANY ('CSE', 'IT');

SELECT * FROM EMP WHERE NOT DEPT = ANY ('CSE', 'IT');

```
SELECT * FROM EMP WHERE NAME LIKE 'A%';
SELECT * FROM EMP WHERE SALARY LIKE '5%';
SELECT * FROM EMP WHERE NAME LIKE 'A%A';
SELECT * FROM EMP WHERE NAME LIKE 'R___';
SELECT * FROM EMP WHERE NAME LIKE '____';

SELECT * FROM EMP WHERE NAME NOT LIKE 'R%';
```

SET operators:-

- The set operators are used to fetch the data from the two tables.

Note:-

- The two tables structure should be same.
- The same number of columns and order of columns must be the same.

UNION ALL:

- IT Fetches (selects) All the records from both the tables including the duplicates

Ex:-

```
SELECT * FROM VOTER UNION ALL SELECT *
    FROM VOTER_LIST;
```

UNION:-

- IT selects only unique Records from both the tables.

Ex:-

```
SELECT * FROM VOTER UNION SELECT *
    FROM VOTER_LIST;
```

SELECT NAME, COURSE FROM STUDENT UNION SELECT NAME, DURATION FROM COURSE;

INVALID QUERIES:

SELECT NAME, AGE FROM VOTER UNION SELECT * FROM VOTER_LIST;

SELECT NAME, AGE FROM VOTER UNION SELECT AGE, NAME FROM VOTER_LIST

SELECT * FROM STUDENT UNION SELECT * FROM COURSE;

INTERSECT

- IT selects common Records (duplicates) from both the tables.

EX:

SELECT * FROM VOTER INTERSECT SELECT * FROM VOTER_LIST;

SELECT NAME FROM VOTER INTERSECT SELECT NAME FROM VOTER_LIST

MINUS:-

IT Selects only UNIQUE Records from the left table. (FIRST TABLE).

EX:

SELECT * FROM VOTER MINUS SELECT * FROM VOTER_LIST;

SELECT NAME FROM VOTER MINUS SELECT NAME FROM VOTER_LIST;

Functions in SQL:-

- SQL has 2 types of functions
  1. Aggregate functions
  2. Scalar function.

AGGREGATE functions:-

- The functions which are working on set of data values and Produce a single value as a result.

Scalar functions:-

- The functions which are working on set of Data values and Produces a result for every data value.

Types in Aggregate functions:-

1. GROUP functions.
2. LIST function.

Types in scalar functions:-

1. Numeric Functions.
2. Character functions.
3. Date function.
4. Conversion function.

AGGREGATE functions:-

1. Group functions:-

- All the group functions work only on columns of the Table.
- SQL Provides 6 group functions such as:

MIN(), MAX(), SUM(), AVG(), COUNT() & COUNT(ALL) / COUNT(*).

MIN():

- It Returns the minimum value of the given column of a Table/view.

Ex:

Select MIN(SALARY) FROM EMP;

Select MIN(SALARY) FROM EMP where DEPT = 'CSE';

Select MIN(SALARY) FROM EMP where DEPT='CSE'
AND ~~DEPT = 'IT'~~
Gender = 'F'.

Select MIN(SALARY) FROM EMP where DEPT = 'IT' AND NAME LIKE 'R%'.

Select MIN(SALARY) FROM EMP where DEPT = 'CSE' OR DEPT = 'IT';

MAX():

- It Returns the maximum value of the given column of a Table/view.

Ex: Select MAX(SALARY) FROM EMP

Select MAX(SALARY) from emp where gender = "M"
AND DEPT = 'IT';

Select MAX(SALARY) FROM EMP where DEPT = 'CSE' AND work_exp >= 5 and Gender = 'M';

Select MIN(SALARY) AS "MINIMUM_SALARY", MAX(SALARY) AS "MINIMUM_SALARY" FROM EMP;

Select MIN(SALARY), MAX(SALARY) from emp where Dept = 'CSE' AND Gender = 'M';

SUM():-

- IT Returns Sum of the given column values of a Table/view.

Ex:- select Sum(Salary) from emp;

Seleld SUM(salary) As "Sum_CSE_MALE" from emp where DEPT='CSE' AND ~~Dept~~ = Gender='M';

select Sum(Salary) from emp where name like 'R%' and dept IN('CSE','IT') or Name like 'A%' and DEPT in ('CSE','IT')

Select Sum(salary) from emp where DEPT='CSE' or DEPT='IT' or DEPT='mech';

AVG():-

- IT Returns An Avgerage value of the given column values of a Table/view.

ex:- Seleld AVG(salary) from emp;

select ~~as~~ AVG(Salary) from emp where gender='M';

Select AVG(salary) from emp where gender='M' and DEPT='CSE';

select AVG(salary) from emp where work_emp >=5 and gender='M';

Count():-

. It Returns count of the given column values of a Table/view.

Ex:- select count(gender) from emp
where gender = 'm';

select count(Dept) from emp where
Dept = Any('CSE', 'IT')

select * from emp where Dept = Any('CSE', 'IT')

Select COUNT(DEPT) from emp where dept IN
('CSE', 'IT') And Gender = 'm'.

Select COUNT(DEPT) from emp DEPT IN('CSE', 'IT')
and work_emp >= 5;

Select count(NAME) from emp where name
like 'A%' and gender = 'm' or Gender = 'm' and
name like 'R%';

select * from emp where name like 'A%'
and gender = 'm' or gender = 'm' and Name like 'R%'.

Count(*):

. It Returns the Total No. of Records in given Table/view.

Ex: select COUNT(*) from emp;

select COUNT(*) AS "TOTAL RECORDS"
from emp;

Select COUNT(*) from emp where
Dept = 'CSE';

select COUNT(*) from emp where Dept = ANY('CSE', 'IT')
and work_emp >= 5;

select ~~count~~ * from emp where Dept = Any('CSE', 'IT')
and work_emp >= 5;

---

select least (56, 76, 34, 52, 3, 8, 94, 57, 31, 25) from dual;

select greatest (56, 76, 34, 52, 3, 8, 94, 57, 31, 25)
from dual;

Numeric functions:

SQRT():
Power():
MOD():
ABS():
FLOOR():
CEIL():
ROUND():
TRUNC():
SIGN():

1. SQRT(): It returns the square root of a given value.

Ex: select SQRT(12) FROM DUAL;

select M1, SQRT(M1) FROM CEC;

2. Power(): It returns the x power of n value.

select power(2,5) from DUAL;

select M1, Power(M1,2) FROM CEC;

3. MOD(): It returns the remainder of the division operation.

Select MOD(10,4) from DUAL;

Select MOD(M1,3) from CEC;

4. ABS(): It returns th. An Absolute value of the given value.

Ex: Select ABS(-5) from DUAL;

Select M1, ABS(M1) from CEC;

5. FLOOR(): It returns the floor value (Nearest small integer value) of the given float value.

Ex: Select floor(12.5346) from dual.

Select Average, FLOOR(Average) from MFC;

6. CEIL(): It returns the ceil value (Nearest large integer value) of the given float value.

Ex: select CEIL(12.5346) from dual.

Select Average, CEIL(Average) from MEC;

7. ROUND(): It rounds the given float value to either ceil or floor value.

Ex: Select ROUND(12.5346) FROM DUAL;

select ROUND(12.45346) FROM DUAL;

Select Average, ROUND(Average) from MEC;

8. TRUNC(): It truncates (Removes) the Fractional Part from the given float value.

Ex: Select TRUNC(12.5346) FROM DUAL.

select Average, TRUNC(Average) from MEC;

9. SIGN():

Select 10/3 from DUAL;

select TRUNC(10/3) from DUAL;

9. select TRUNC(39.596785,2) from DUAL;

9. select Average, TRUNC(Average, 2) from Mec;

9. SIGN:- IT Returns 0 If the number is a zero
returns 1 iff the number is a positive or
returns -1 if the number is negative.

Ex: select SIGN(5) from DUAL;

select SIGN(-5) from DUAL;

select SIGN(0) from DUAL;

select MI, SIGN(MI) from Mec;

String or character functions:-

Length():
INITCAP():
LOWER():
UPPER():
Concat():
Replace():
Reverse()
SUBSTR():
INSTR():
TRAN

1. Length (): It returns the length of the given string.

Ex:
Select length ('welcome to SQL Functions') from DUAL;
Select NAME, length (NAME) from emp;
select SALARY, length (Salary), length (work_emp) from emp;

2. INITCAP():
It returns capitalize the first characters of the words of given string.

Ex: Select INITCAP ('welcome to sql function') from DUAL;
select NAME, INITCAP (NAME) from emp;

3. Lower(): It converts all the upper case characters to lower case.

Ex: Select LOWER ('WELCOME TO SQL') from DUAL;
select NAME, LOWER (NAME) FROM EMP;
Select NAME, INITCAP(NAME), LOWER (NAME) FROM EMP.
Update EMP SET NAME = INITCAP (NAME);

select * from CEC;

Insert into CEC VALUES (INITCAP ('&NAME'), &M1, &M2, &M3, &TOTAL)

4. Upper(): It converts all the lower case characters to upper case:

Ex: Select UPPER ('welcome to SQL') from DUAL;
Select NAME, UPPER (NAME) from emp;

5. Concat():

It concatenates one string at the end of another string.

Ex: select CONCAT ('welcome to', 'SQL function') from DUAL;

~~select concat~~

select NAME, CONCAT ('HELLO...', NAME) from CEC;

Replace():

It Replaces a given string with another string.

Ex:

Select Replace ('welcome to SQL functions', 'SQL', 'PL/SQL') FROM DUAL;

Select Replace ('WELCOME', 'E', 'A') FROM DUAL;

Select NAME, REPLACE (NAME, 'A', 'Y') FROM VOTER;

Reverse():

. It Reverse the given string.

Ex:- SELECT REPLACE ('WELCOME') FROM DUAL;

SELECT NAME, REVERSE (NAME) FROM VOTER;

SUBSTR(). [SUBSTR (STRING, START, length)]

. It Returns a substring in the given string based on the given values.

Ex: select SUBSTR ('WELCOME TO SQL Functions', 12, FROM DUAL;

Select SUBSTR ('WELCOME TO SQL Functions', 12, 3)
FROM DUAL;

select NAME, SUBSTR (NAME, 3) FROM VOTER;

INSTR(): [INSTR(String, SubSTR, Direction, occurance)]

. It returns the index of the substring in the given string.

EX:-

Select INSTR ('WELCOME To SQL Functions', 'E')
from DUAL;

Select INSTR ('WELCOME TO SQL Functions', 'E', '1')
From DUAL;

Select INSTR ('WELCOME To SQL Functions', 'E', '-1')
From DUAL;

. The direction value may be either 1 or -1.
The default value is 1.

. If the direction value is 1, then the substring searches from left to right.

. If the direction value is -1, then the substring searches from right to left.

. '1' gives the first occurance but the '-1' gives the last occurance of the substring.

SQL> Select INSTR ('WELCOME To SQL Functions', 'E', 1, 1)
from DUAL;

Select INSTR ('WELCOME TO SQL Functions', 'E', 1, 2)
from DUAL;

Select INSTR ('WELCOME TO SQL Functions', 'E', -1, 2)
from DUAL;

TRANSLATE():- [Similar to Replace function]

- It Replaces the given string character by character.

Ex:-
```
Select Translate('WELCOME To SQL Functions',
                 'COME', 'ABCD')
from DUAL;
```

ASCII():-

- It Returns ascii value of the given character

Ex:-
```
Select ASCII('A') from DUAL;
select ASCII('a') from DUAL;
```

CHR():-

- It Returns a character of the given Ascii value

Ex:- Select CHR(65) from DUAL;

TRIM():-

- It ~~Returns~~ Removes the begining and ending white spaces of the String.

Ex:-
```
Select length(' WELCOME ') from DUAL;
select length(' WELCOME') from DUAL;
Select length(' WELCOME '),
       length(TRIM(' WELCOME ')) from DUAL;
```

RTRIM():-

- It Removes the while spaces at ending of the string

Ex:

```
Select Length (' WELCOME ') from DUAL;
select length (' WELCOME '),
       length (RTRIM (' WELCOME ')) from dual;
```

LPAD():-

Ex:-

```
Select  LPAD ('WELCOME', 15) from DUAL;
select  LPAD ('WELCOME', 15, *) from DUAL;
```

RPAD():-

```
Select  RPAD ('WELCOME', 15) from DUAL;
select  RPAD ('WELCOME', 15, *) from DUAL;
```

Conversion functions():-

```
Select  TO_CHAR(10) from DUAL;
Select  To_Number ('10') from DUAL;
Select  To_DATE ('12-AUG-2020') from dual;
```

Date functions ():-

- It returns the current date of the system.

EX:-

```
Select sysdate from dual;
select Add_Months (sysdate,10) from DUAL;
select Add_Months ('12-May-2022',10) from dual;
Select Months_between (SYSDATE, '12-AUG-2020') from dual;
select Next_Day (sysdate, 'wed') from dual;
Select Last_Day (sysdate) from DUAL;
```

Unique constraint

- IT cannot accept duplicate values into the colu
- IT can accept NULL values into the column.
- IT is a Table level constraint, so we can apply at column level and table level also.

↓

unique constraint:

Create:

Column Level:

```
create table empool (Eid number(3) NOT NULL unique,
Name varchar(10) NOT NULL, Dept varchar(10),
Gender char(1), Salary number(5),
mobile number(5) unique);
```

Named constraint:

```
create table voter (Id number(3) constraint
unique_id unique, name varchar(10) NOT NULL,
gender char(1) NOT NULL, age number(3),
address varchar2(10));
```

NOT NULL & unique:

```
create table voter (id number(3) constraint
unique_id NOT NULL UNIQUE, Name varchar(10)
not null, age number(3), address varchar2(10));
```

Table level:

```
create table empool (eid number(3), NOT null,
name varchar(10) not null, dept varchar(10), gender char(1)
Salary number(5), mobile number(10), unique(Id, mobile));
```

create table voter(id number(3), name varchar(10) not null, gender char(1) not null, age number(3), address varchar(10), mobile number(10), unique(Id, mobile));

Named Constraint:

create table voter(id number(3), name varchar(10) not null, gender char(1) not null, age number(3), address varchar(10), mobile number(10), constraint unique_id_mobile unique(ID, mobile));

After Command Constraint:

create table empool (eid number(3), name varchar(10), gender char(1), dept varchar(5), salary number(5), mobile number(5), unique(eid, number));

create table voter (id number(3), name varchar(10), mobile number(10));

Alter table voter add constraint unique_id unique(id);

Alter Table voter add constraint id_mobile_unique unique (id, mobile);

Alter table voter modify id unique.

Alter table voter modify mobile constraint mobile_unique unique;

voter

Alter table drop constraint id_mobile_unique.

Primary key:- [Combination of not null & unique]

- It is a combination of not null and unique constraints.
- It cannot accept null and duplicate values into the column.
- Primary key constraint is useful to uniquely identify the records of a table.
- A Table can have only one primary key but we can apply the primary key on multiple columns [that is called composit primary key].
- It is a table level constraint.

Ex:- Create table voter (id number(3) primary key, name varchar(10), mobile number(10));

Create table voter (id number(3) constraint id_pk primary key, name varchar(10), mobile number(10));

Table level:-

Create table voter (ID number(3), name varchar(10) not null, mobile number(10), primary key(id));

Create table voter (ID Number(3), NAME varchar(10) not null, mobile number(10), primary key (id, mobile));

Create table voter (id number(3), name varchar(10) not null, mobile number(3), constraint pk_key primary key (id, mobile));

Alter:-

Alter table voter add Primary key (id);

Alter table voter modify id Primary key;

Alter table voter add constraint ID-Pk Primary key (id, mobile);

Alter table voter drop Primary key;

Alter table voter drop constraint Pk-key.

---

sql> set linesize 120;

→ It will set column lines to 120.

Check Constraint:-

- The check constraint checks whether the values of a column is satisfying a particular condition or not. If the value is satisfying the given condition then only the records will be inserted into the table otherwise it rejects.

Ex:- Create Table VoterID number(3) Primary key,
Name varchar(15) not null,
Gender varchar (1) not null,
Age number (3) check (Age>18),
Address varchar(15));

Named constraint:-

Create Table VoterCID number(3) Primary key,
Name varchar (10) Not null,
Gender varchar (1) Not null,
Age number(3) constraint check_Age check (Age>18),
Address varchar (10));

Table level constraint:-

```
Create table CEC(Name varchar(10), M1 number(3),
M2 number(3), M3 number(3), Total number(3),
Avg number(5,2),
Check(M1>=0 AND M2>=0 AND M3>=0 AND
M1<=100 AND M2<=100 AND M3<=100));
```

Named Constraint:-

```
Create table cec (name varchar(10), M1 number(3),
M2 number(3), M3 number(3), Total number(3),
Avg number(5,2), constaint marks_check.
Check (M1>=0 AND M2>=0 AND M3>=0 AND
M1<=100 AND M2<=100 AND M3<=100));
```

using Alter:-

```
Alter table voter Add (Primary key(Id), check(Age>=19));
Alter table cec add check(M1>=0 AND M1<=100);
Alter table voter Modify (Id Primary key, Age
                          check(Age>=19));

Alter table cec add constraint M2_chk check(M2>0
                                AND M2<=100);

Alter table drop constraint M2_chk;

Alter table cec modify M3 check(M3>=0
                          And M3<=100);
```

Select constraint_name from user_constraint where table_name = 'voter';

Alter table voter add (constraint pk_id Primary key(id), constraint age_chk check (Age >= 19));

Alter table voter modify (constraint pk_id primary key(id), constraint age_chk check (Age >= 19));

Default constraint:

. The default constraint is use to Provide a default value to the column.

Create:

create table voter (id number(3) Primary key, name varchar(10), gender char(1), Age number(3) check (age >= 19), Address varchar(15) default 'Guntur');

Alter:

Alter table voter modify Address default 'Guntur';

Alter table voter modify gender default 'M';

Alter table voter add Address varchar(10) default 'GUNTUR';

01-09-2022

SQL:-

Foreign key constraint:-

- Foreign key constraint is used to establish the link b/w the two tables.
- The foreign key of a table always refers to the primary key of another table.
- A table can have only one primary key but a table can have one or more foreign keys.
- Primary key can't accept duplicate and null values in the column but a Foreign key can accept.
- To delete a record from the master table, first delete that record in the child table and later delete that record from the master table.
- To drop [Delete] the master table, first drop the child table then drop the master table.

Master table

```
create table course (Name varchar(10) primary key,
                     Duration varchar(10) not null,
                     Fee number(5));
```

Child table:

```
create table Student (Id number(3) primarykey,
                      Name varchar(10) NOT null,
                      Gender char(1) Default 'M', course varchar(10)
                      Course varchar(10) References course (name),
                      Mobile number(5) not null);
```

With name:

```
create table Student (id number(3) primary key,
    name varchar(10), course varchar(10)
    constraint FK_course references course(name),
    mobile number(5));
```

At table level:

```
create Table Student (id number(3) Primary key,
    name varchar(10) NOT null, Gender char(1)
    default 'M', course varchar(10), mobile number
    (5) not null, foreign key (course)
      References course (name));
```

With name:-

```
create Table Student (id number(3) Primary key,
    name varchar(10), course varchar(10), mobile number(5),
    constraint F_key Foreign key (course) References
      course (name));
```

Alter:

Without Name:-

```
Alter table student add foreign key (course)
                References course (name))

Alter table Student add constraint F_key
    Foreign key (course) References course (name);

Alter table student modify course References
                course (name);
```

* [Alter table student modify constraint F_key
foreign key (course) References course (name);]

Note:- It is not possible by named constraint.

Synonyms in sql:

- In sql, a Synonym is a permanent alias name to the database object.

Syntax:

```
Create Synonym synonym_name for [Any_]object_name;
```

Ex: Create Synonym x for student;

Drop Synonym:

Syntax:

```
Drop synonym synonym_name;
```

Ex: Drop synonym x;

Views in sql:

- A view is an imaginary or a virtual table.
- Views can be used to hide the sensible data of the tables.
- We can create two types of views such as:
  1. Simple or updatable or rewritable views:
  2. Complex or readable views:

Simple view:

- If a view was created based on the data of one Table.
- On simple views, we can perform All DML operations.

- Simple views are updatable views, means that, if we update the table then the view was updated and if the view was updated
- on simple views, we can insert, update and delete the records and as well as we can read the data.

Syntax:

Create [OR Replace] view view_name as select Column_List from Table_name [where condition];

Ex:

Create or replace view CSE-Emp AS Select * from emp where dept = 'CSE';

Select * from CSE-emp;

insert into emp values(6, 'Kiran', 'CSE', 'M', 46000, 1, 7, 4500);

create or replace view csE_emp AS select name, dept, gender, salary from emp where dept = 'CSE'.

Complex views:

- If a view was created based on the data of two or more table.
- complex views are readable views only. we can't insert the records into the complex views. But sometimes we can perform updation and deletion operations.

Syntax:

create or replace view view_name as Select Table1.column1, Table1.column2, ... from Table1, Table2 where condition;

Create or replace view course_student as select Id, Student.name AS "S_name", gender, course.name AS "C_name", Duration, Fee, mobile from course, student where course.name = student.name.

Select * from course_student;

Joins in SQL:-

- Joins are used to fetch the data from two or more tables.
- Based on the common column of the table, we can join the tables.
- In SQL, joins are categorized as:
  1. Inner or Equi Join.
  2. outer or non-equi Join.
     - Left outer Join
     - Right outer Join
     - Full outer Join
  3. Self Join.
  4. Cross Joins.

Equi / Inner Join:

- It returns the records from both the tables with records are satisfying the specified joining condition.

Ex:-

Select student.name, course.name, fee, mobile from student inner join course on student.course = course.name;

outer / non-equi join:-

1. **Left join:-**
   - It returns all the records from the left table and only the matched records from the right table.

Ex:- Select student.name, cause.name, fee, mobile from course left join student on student.course= course.name;

2. **Right join:-**
   - It returns all the records from the right table and only the matched records from the left table.

Ex:- select student.name, course.name, fee, mobile from course right join student on student.course = course.name;

3. **Full join:-**
   - It returns all the Records from Both the table.

Ex:- Select student.name, course.name, fee, mobile from student full join course on student.course= course.name;

* **cross join:-**

Ex:- select student.name, course.name from student cross join course;

Self join:-

Ex:- Select T1.name, T2.name from CEC T1, CEC T2 where T1.m1 >= T2.m2;

DCL command (data control language):

1. Grant:

It is used to grant the permissions to the clients.

Ex:

Grant, Insert, select on employee to user_name/public;

2. Revoke:

It is used to take back (Revoke) the granted permissions from the clients.

Ex: Revoke insert, select on employee from user_name/public;

Indexes in SQL:

- An Index is like a table consisting of two columns.
  1. one is row-id and other one is value of the column on which column we created an index.

Table

ID	Name	Gender	Salary	Mobile

INDEX

Row-ID	value

Types of index

Simple Index
Composit Index
unique Index
Reverse Index.

Some syntax for all :-

Create [unique] Index index_name on Table_name
(column_name) [Reverse];

. The oracle search engine (Tool) uses the indexes to speed up the searching process.

1. Simple Index:-
. Index created on one column of a Table.

2. Composit Index:-
. Index Created on two or more column of a Table.

3. unique Index
. Index created on unique/primary key column of a Table.

4. Reverse Index:-
. Index created on one or more column of a Table with reverse clause.

Ex: Create Index sal_index on emp(salary); - simple.

create Index Gender_sal_index on emp(gender, salary)
- Composit Index.

Create unique index EID_Index on emp (Salary)
- - unique

Create index sal_index on emp(salary) reverse;

-- Reverse index.

How to see index in a table:-

Select index_name from user_indexes where table_name = 'emp';

How to drop Index:-

Drop index sal_Index;

TCL (Transaction control language):-

- DDL commands are directly interacts with the Database Server and the data is automatically saved into DB that means implicite commit.
- DML Commands, there is no direct communication to the Data Base Server. the data is temporarily stored in ram. when you exit the / Cancle the session then data is Stored into the Server.

- Commit — syntax :- commit;
- Rollback — Syntax :- Rollback;
- Save point — syntax :- save point p1;

Sequences in SQL:-

- Sequence is a Database, which is used to generate random numbers.

Syntax:

```
create sequence sequence_Name [START with value
                Increment by value.
                minvalue value | NOMinvalue
                Maxvalue value | NOMaxvalue
                cycle | Nocycle
                Cache value | No cache];
```

Ex:- create sequence S1; | 1 To infinity

Insert into voter values (S1.nextval, 'name')

create sequence S1 start with 101;
-- from 101 to infinity

create sequence S1 stat with 101
increment by 10;

Drop sequence S1;
Insert into voter values (S1.Nextval, 'name')

Create sequence S1 start with 101
increment by 10 Maxvalue 150 cache 5;

Create sequence S1 start with 101 increment by
10 Maxvalue 150 cycle cache 5;

Create Sequence S1 start with 101 increment by 10
minvalue 10 maxvalue 150 cycle cache 5;

→ By default cache value is 20

Insert into voter values (concat('GNT',
TO_char (s1.nextval)), 'A NAME');

ID	NAME
GNT10001	RAVI
GNT 10002	Rani
GNT10003	Poornima
GNT 10004	Nikhil
GNT 10005	Seenu

Sub / Nested / inner queries:

dept
cn , eid , max sal

select Name, dept from emp where salary =
(select max (salary) from emp);

Types:

1. Non-correlated sub queries.
2. Correlated sub queries.

1. First inner query will be executed later the outer query will be executed.

Sub queries/Nested queries/Inner queries:-

→ A query consisting another query is called sub query.

Based on the execution of order the queries are classified into two types.

Types:

1. Non-correlated sub queries:
   - First the inner query executes later outer query.
     - Single value/Row:- It returns single value/row.
     - Multi value/Row:- It returns multi value/rows.

2. Co-related sub query:-
   - ~~First query~~
   - First outer query will execute then inner query again the outer query.
   - We can use EXISTS & NOT exists operators.

Single value/Row:-

- We can use all relational operators like [<, <=, =, !=, >, >=]

Multi value/Row:

- We can use any, All, In, Not any, Not all, NOT IN

Single Value/Row:-

Ex:-

```
Select ename, job from emp where
    Sal = (select MAX(Sal) from emp);

Select ename from emp where Sal =
(select MAX(sal) from emp where Job = 'SALESMAN');
```

Multi Value/Row:-

Ex:-

```
Select * from emp where Ename = Any
(Select Ename from emp where Job like 'S%' or
    Job like 'A%');

select * from emp where ename IN
(select ename from emp where Job like 'S%'
  or Job like 'A%');

select * from emp where Not ename = Any
(select ename from emp where Job like 'S%'
  or Job like 'A%');
```

Co-related sub queries:-

Table course:-

```
Create table course (Name varchar (10)
    Primary key, Duration varchar(10),
                Fee number (5));
```

Table student:-

```
Create table Student (Id number(3) Primary key,
name varchar (10), Gender char(1), course varchar (10)
References course (Name), Mobile number (5));
```

EX:-

```
select name from course where NOT exists
(select course from student where Studen.course=
                course.name);
```

↓

Anti joins.

```
select name from course where exists
(select course from student where
    student.course = course.Name);
```

```
SQl> select Dname from dept where NOT exists
    (select deptno from emp where
        emp.deptno = dept.deptno);
```