Date: 26-01-2022:
===============
Introduction to SQL:
=================
- SQL stands for "Structure Query Language".
- introduced by IBM for "communicate with database".
- called as "Sequel" renamed into "SQL"
- SQL queries standered query.SQL language queries cannot be case sensitive.

    ex:

```
select * from emp;
SELECT * FROM EMP;
SeLect * fRom emp;
```

- Sub-languages of SQL:

1. Data Definition Language(DDL):
==============================
- Create
- Alter
- Rename
- Truncate
- Drop

New commands:
=============
- Recyclebin
- Flashback
- Purge

2. Data Manipulation Language(DML):
================================
- Insert
- Update
- Delete

New command:
=============
- merge------------------> JOINS
- insert all------------> DQL COMMAND

3. Data Query / Retrieval Language(DQL / DRL):
=======================================
- Select

4. Transaction Control Language(TCL):

================================

- Commit
- Rollback
- Savepoint

5. Data Control Language(DCL):

===========================

- Grant
- Revoke

===
=====================

Datatypes in Oracle:

=================

What is datatype:

==============

- is an attribute which is used to store the type of data into a column.

1. Numeric datatypes
2. Character / String datatypes
3. Long datatypes
4. Date datatypes
5. Raw & Long Raw datatypes
6. Lob datatypes(Large objects datatypes)

1. Numeric datatypes:

==================

i) INT
ii) NUMBER(P,S)

i) INT:

======

- store integer format values only.
- when we use "int" datatype on a column at the time of table creation internally oracle db server is converting into "number(38)".

================
|| int = number(38) ||
================

ii) NUMBER(P,S):

==============

- can store integer and also float values into a column.

number(p) : storing integer values only.
number(p,s) : storing float values only.

Precision(P):

- counting all digits including left & right sides of digits of given float expression.

ex:

```
56.23
precision = 4

7346.27
precision = 6
```

Scale(s):

- counting the right side digits only.

ex:

```
56.23
scale = 2
precision = 4

75683.123
scale = 3
precision = 8
```

2. Character / String datatypes:

- storing string format data only.
- string is represent with 'string'.

```
                      string format data
                              ||
              characters only                alphanumeric
              string format data             string format data
                      ||                     ||
              ( A - Z (or) a - z )           ( A - Z (or) a - z
                                             &
                                           0  ------------ 9
                                             &
                                      @,#,$,%,_,&,........ etc )

     ex: 'SMITH' , 'smith',.......etc        ex: 'sai123@gmail.com',...etc
```

- it again two types those are

1. Non - Unicode datatypes
========================
- to store localized data(supporting only english lang.)

  I) char(size)
  II) varchar2(size)

2. Unicode datatypes:
==================
- to store globalized data (all languages i.e National lang's)

  I) Nchar(size)
  II) Nvarchar2(size)
- "N" stands for National.

I) char(size):
==========
- fixed length datatype(static datatype)
- 1 char = 1 byte
- max size is 2000 bytes.

ex:
Empname char(2000)
===============
'SAI'---------> SAI --------------------> 3 bytes - 10 bytes = 7 bytes wasted
'WARD'----> WARD ------------------> 4 bytes - 10 bytes = 6 bytes wasted
'2000chars name' ----> allowed
'2001chars name'-----> error

I) varchar2(size):
==============
- variable length datatype(dynamic datatype)
- 1 char = 1 byte
- max size is 4000 bytes.

ex:
Empname varchar2(10)
==================
'SAI'---------> SAI --------------------> 3 bytes - 10 bytes = 7 bytes saved

```
        'WARD'---->  WARD ------------------> 4 bytes - 10 bytes = 6 bytes saved
'2000chars name' ----> allowed
'2001chars name'-----> error
```

I) Nchar(size):
==========

```
        - fixed length datatype(static datatype)
        - 1 char = 1 byte
        - max size is 2000 bytes.

        ex:
                Empname Nchar(2000)
                ===============
        'SAI'--------->  SAI ---------------------> 3 bytes - 10 bytes = 7 bytes wasted
        'WARD'---->  WARD ------------------> 4 bytes - 10 bytes = 6 bytes wasted
'2000chars name' ----> allowed
'2001chars name'-----> error
```

II) Nvarchar2(size):
==============

```
        - variable length datatype(dynamic datatype)
        - 1 char = 1 byte
        - max size is 4000 bytes.

        ex:
                Empname Nvarchar2(10)
                ===================
        'SAI'--------->  SAI ---------------------> 3 bytes - 10 bytes = 7 bytes saved
        'WARD'---->  WARD ------------------> 4 bytes - 10 bytes = 6 bytes saved
'2000chars name' ----> allowed
'2001chars name'-----> error
```

3. Long datatype:
================

- to store Unicode and also Non-unicode char's.
- 1 char = 1 bytes
- dynamic datatype
- max size 2 gb.

4. Date datatypes:
===============

- store date & time information.
- range of date is from '01-jan-4712 bc' to '31-dec-9999 ad'

i) date :
======
- storing date & time information of a particular day.
but time is optional.if user is not enter time then oracle server will take the time '12:00:00am' (or) '00:00:00am'.
- default date format of oracle : 'dd-mon-yy/yyyy hh:mi:ss '

ex: '28-jan-2022 / 22 19:54:SS '

- this datatype is occupied 7 bytes memory(fixed memory)

ex:
```
        ' DD - MON - YYYY/YY   HH:MI:SS '
           1      1        2           1   1     1
```

ii) timestamp:
============
- storing date & time information along with milliseconds.
- occupied 11bytes memory(fixed memory)

ex: '28-jan-2022/22   19:59:12.1000'

```
ex: 'DD - MON - YYYY / YY    HH:MI:SS.MS '
      1       1                2          1     1   1   4
```

5. Raw & Long Raw datatypes:
========================
- store binary format data(0100101010101010) (i.e image / audio / video).
  - Raw -----------> 2000 bytes
  - Long Raw----> 2gb

6. Lob datatypes(Large objects datatypes):
===================================
i) Clob :
  - character large object
  - store non-unicode char's

- char(size) ---------- 2000 bytes
- varchar2(size) --- 4000 bytes
- Long ----------------- 2gb

- Clob ------------------ 4gb

ii) NClob :
  - National character large object
  - store unicode char's

- Nchar(size) ---------- 2000 bytes
- Nvarchar2(size) --- 4000 bytes
- Long ------------------ 2gb
- NClob ------------------ 4gb

iii) Blob:
  - Binary large object
  - store binary format data 0101010101010001010(image/audio/video)
- Raw ---------------- 2000 bytes
- Long Raw ------- 2gb
- Blob --------------- 4gb

===

1. Data Definition Language(DDL):
==============================
- Create
- Alter
- Rename
- Truncate
- Drop

New commands:
=============
- Recyclebin
- Flashback
- Purge

Create:
======
- create a new table in oracle db.
  (or)
- create a new db object in oracle (new db objects are :
tables,views,synonyms,index,clsuter,sequence,
  - procedure,functions,triggers ,package,.......etc)

syntax to create a new table in oracle:

=================================
create table <table name>(<column name1> <datatype>[size],<column name2>
<datatype>[size],......);upto 1000col

ex:
SQL> CREATE TABLE STUDENT(STID INT,SNAME CHAR(10),SFEE NUMBER(6,2));

TO VIEW THE STRUCTURE OF A TABLE :
==================================
SYNTAX:
========
DESC <TABLE NAME>; (DESCRIBE)

EX:
SQL> DESC STUDENT;

Alter:
=====
- TO CHANGE THE STRUCTURE OF A TABLE.
- SUB LANGUAGES OF ALTER COMMAND.
    i) ALTER - MODIFY
    ii) ALTER - ADD
    iii) ALTER - RENAME
    iv) ALTER - DROP

i) ALTER - MODIFY:
================
- TO CHANGE DATATYPE AND ALSO THE SIZE OF DATATYPE OF
A PARTICULAR COLUMN IN A TABLE.

SYNTAX:
========
ALTER TABLE <TN> MODIFY <COLUMN NAME> <NEW DATATYPE>[NEW SIZE];

EX:
SQL> ALTER TABLE STUDENT MODIFY SNAME VARCHAR2(20);

ii) ALTER - ADD:
==============
- TO ADD A NEW COLUMN TO AN EXISTING TABLE.

SYNTAX:
=======

ALTER TABLE <TN> ADD <NEW COLUMN NAME><DATATYPE>[SIZE];

EX:
SQL> ALTER TABLE STUDENT ADD SADDRESS VARCHAR2(20);

iii) ALTER - RENAME:
==================
- TO CHANGE A COLUMN NAME IN TABLE.

SYNTAX:
========
ALTER TABLE <TN> RENAME <COLUMN> <OLD COLUMN NAME> TO <NEW COLUMN NAME>;

EX:
SQL> ALTER TABLE STUDENT RENAME COLUMN SNAME TO STUDENTNAMES;

iv) ALTER - DROP:
===============
- TO DROP / DELETE A COLUMN FROM A TABLE.

SYNTAX:
=======
ALTER TABLE <TN> DROP <COLUMN> <COLUMN NAME>;

EX:
SQL> ALTER TABLE STUDENT DROP COLUMN SFEE;

- Rename:
=========
- to rename a table name.

syntax:
=======
RENAME <OLD TABLE NAME> TO <NEW TABLE NAME>;

EX:
SQL> RENAME STUDENT TO SDETAILS;
SQL> RENAME SDETAILS TO STUDENT;

Truncate:
=========
- TO DELETE ALL ROWS FROM A TABLE(BUT COLUMNS)
- TO DELETE A SPECIFIC ROW IS NOT POSSIBLE-------> NOT SUPPORTS

"WHERE" CONDITION.

SYNTAX:
========
TRUNCATE TABLE <TABLE NAME>;

EX:
SQL> TRUNCATE TABLE STUDENT;

NOTE : ROWS DATA IS PERMANENT DELETION.(WE CANNOT RESTORE)

Drop:
=====
- TO DROP / DELETE A TABLE(ROWS & COLUMNS) FROM DATABASE.

SYNTAX:
========
DROP TABLE <TABLE NAME>;

EX:
SQL> DROP TABLE STUDENT;
Table dropped.

NOTE:
=====
- BEFORE ORACLE10g ENTERPRISE EDITION ONCE WE DROP A TABLE
i.e PERMANENT.BUT FROM ORACLE10g ENTERPRISE EDITION ONCE WE DROP A TABLE
i.e TEMPORARY.

Q. DROPPED TABLE IS WHERE IT IS?
A: RECYCLEBIN:
==============
- IS A PRE-DEFINED / SYSTEM DEFINED TABLE.
- IS USED TO STORE THE INFORMATION ABOUT DROPPED TABLES FROM DB.

TO VIEW THE STRUCTURE OF RECYCLEBIN:
======================================
SYNTAX:
========
DESC RECYCLEBIN;

EX:

```
SQL> DESC RECYCLEBIN;
```

HOW TO VIEW DROPPED TABLES INFORMATION IN RECYCLEBIN:
===
SYNTAX:
=======

```
SQL> SELECT OBJECT_NAME,ORIGINAL_NAME FROM RECYCLEBIN;

OBJECT_NAME                          ORIGINAL_NAME
------------------------------------ ----------------------------
BIN$1ZDEaBi+QsKUcEXZJ738yg==$0       STUDENT
```

Q. HOW TO RESTORE DROPPED TABLE?
A: FLASHBACK:
=============

- IT IS DDL COMMAND
- IS USED TO RESTORE A TABLE FROM RECYCLEBIN.

SYNTAX:
========

```
FLASHBACK TABLE <TABLE NAME> TO BEFORE DROP;
```

EX:

```
SQL> FLASHBACK TABLE STUDENT TO BEFORE DROP;Flashback complete.
```

Q. HOW TO DROP TABLE PERMANENTLY?
A: PURGE:
=========

- it is ddl command
- which is used to drop a table permanently.

syntax1: to drop a specific table from recyclebin permanently:
===
syntax:
=======

```
    purge table <table name>;
```

ex:

```
SQL> PURGE TABLE TEST1;

Table purged.
```

SYNTAX2: to drop all tables from recyclebin permanently:
==
SYNTAX:
========
PURGE RECYCLEBIN;

SYNTAX3: to drop a table from database permanently:
==
syntax:
=======
drop table <table name> purge;

ex:
SQL> DROP TABLE TEST55 PURGE;

2. Data Manipulation Language(DML):
===============================
- Insert
- Update
- Delete

New command:
=============
- merge
- insert all

Insert:
======
- TO INSERT A NEW ROW DATA INTO A TABLE.

SYNTAX1:
=========
INSERT INTO <TN> VALUES(VALUE1,VALUE2,............);

EX:
SQL> CREATE TABLE EMP(EID INT,ENAME VARCHAR2(10),SAL NUMBER(10));
SQL> INSERT INTO EMP VALUES(1,'SMITH',25000);

SYNTAX2:
=========
INSERT INTO <TN>(<COLUMN NAME1>,......) VALUES(VALUE1,..........);

EX:

```
SQL> INSERT INTO EMP(EID,ENAME,SAL)VALUES(2,'JONES',12000);
SQL> INSERT INTO EMP(EID,ENAME) VALUES(3,'ALLEN');
SQL> INSERT INTO EMP(EID) VALUES(4);
```

HOW INSERT NULLS INTO A TABLE:
=============================
SYNTAX1:
EX:

```
SQL> INSERT INTO EMP VALUES(NULL,NULL,NULL);
```

SYNTAX2:
EX:

```
SQL> INSERT INTO EMP(EID,ENAME,SAL) VALUES(NULL,NULL,NULL);
```

SUBSTITUTIONAL OPERATORS:
===========================

- INSERT MULTIPLE ROWS DATA INTO A TABLE CONTINUE.

i) & : INSERTING VALUES INTO COLUMNS DYNAMICALLY

ii) && : INSERTING VALUES ARE FIXED(WE CANNOT CHANGE).
IF WE WANT TO CHANGE VALUES THEN WE SHOULD EXIT FROM ORACLE.

EX:

```
SQL> INSERT INTO EMP VALUES(&EID,'&ENAME',&SAL);
Enter value for eid: 6
Enter value for ename: MILLER
Enter value for sal: 15000
```

/ (TO RE EXECUTE THE LAST EXECUTED SQL QUERY IN SQLPLUS EDITOR)

```
Enter value for eid: 7
Enter value for ename: SAI
Enter value for sal: 42000
```

EX:

```
SQL> INSERT INTO EMP(EID,ENAME)VALUES(&EID,'&ENAME');
Enter value for eid: 9
Enter value for ename: YUVIN

/
Enter value for eid: 10
Enter value for ename: ADAMS
```

Ex:

```
SQL> INSERT INTO EMP VALUES(&EID,'&ENAME',&&SAL);
Enter value for eid: 14
Enter value for ename: KLM
Enter value for sal: 9000

/
Enter value for eid: 15
Enter value for ename: AWS
```

Update:
=======

- TO UPDATE ALL ROWS DATA IN A TABLE AT A TIME.

  (OR)

- TO UPDATE A SPECIFIC ROW DATA IN A TABLE BY USING "WHERE" CONDITION.

SYNTAX:
=======

```
UPDATE <TN> SET <COLUMN NAME1>=<VALUE1>,<COLUMN
NAME2>=<VALUE2>,..........[WHERE <CONDITION>];
```

EX:

```
SQL> UPDATE EMP SET EID=101,ENAME='BHUVIN',SAL=1100 ;
```

EX:

```
SQL> UPDATE EMP SET JOB='HR',SAL=7500 WHERE EMPNO=7788;
```

EX:

```
SQL> UPDATE EMP SET MGR=1122 WHERE EMPNO=7839;
```

EX:

```
SQL> UPDATE EMP SET SAL=NULL;
```

EX:

```
SQL> UPDATE EMP SET SAL=5000 WHERE SAL IS NULL;
```

Delete:
======

- to delete all rows data from a table at a time.

  (or)

- to delete a specific row data from a table by using "where" condition.

SYNTAX:
========
DELETE FROM <TN> [WHERE <CONDITION>];

EX:
SQL> DELETE FROM EMP WHERE EMPNO=7788;

EX:
SQL> DELETE FROM EMP WHERE COMM IS NULL;

EX:
SQL> DELETE FROM EMP;

TRUNCATE vs DELETE:
===================

TRUNCATE	DELETE
1. CANNOT DELETE A SPECIFIC ROW	1. WE CAN
2. NOT SUPPORTING "WHERE" CONDITION	2. SUPPORTING "WHERE"
3. DATA DELETED PERMANENTLY.	3. DATA DELETED TEMPORARY
4. WE CANNOT RESTORE DATA BY USING "ROLLBACK"	4. WE CAN RESTORE DATA BY USING "ROLLBACK" COMMAND
5. EXECUTION SPEED IS FAST (ALL ROWS WILL MAKE ONE PAGE)	5. EXECUTION SPEED IS SLOW (DELETING ROWS IN ONE BY ONE ROW)

NOTE:
======
- FLASHBACK : RESTORE TABLE INTO DB --------> RECYLEBIN TABLE
- ROLLBACK : RESTORE DATA INTO TABLE ----> INSTANCE MEMORY(RAM)

3) DQL / DRL :
===========

SELECT :
========
- to retrieval / read all rows data from a table.
(or)
- to retrieval / read a specific row data from a table by using "where" condition.

syntax:
======
```
select  * / <list of columns> from <table name> [ where <condition> ];
```

Ex:
```
SQL> SELECT * FROM EMP;
          (OR)
SQL> SELECT EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO FROM EMP;
```

EX:
```
SQL> SELECT EMPNO,ENAME,SAL FROM EMP;
SQL> SELECT * FROM EMP WHERE EMPNO=7788;
SQL> SELECT EMPNO,HIREDATE FROM EMP WHERE EMPNO=7788;
```

EX:
```
SQL> SELECT * FROM EMP WHERE JOB='MANAGER';
SQL> SELECT * FROM EMP WHERE COMM IS NULL;
SQL> SELECT * FROM EMP WHERE COMM IS NOT NULL;
```

HOW TO CREATE A NEW TABLE FROM AN EXISTING TABLE(OLD TABLE):
===
SYNTAX1: A NEW TABLE WITH COPY OF ALL ROWS & COLUMNS FROM OLD TABLE:
==
```
CREATE TABLE <NEW TABLE NAME> AS SELECT * FROM <OLD TABLE NAME>;
```

EX:
```
SQL> CREATE TABLE NEWDEPT AS SELECT * FROM DEPT;
```

SYNTAX2: A NEW TABLE WITHOUT COPY OF ALL ROWS(DATA) FROM OLD TABLE:
==
```
CREATE TABLE <NEW TABLE NAME> AS SELECT * FROM <OLD TABLE NAME> WHERE
<FALSE CONDITION>;
```

EX:
```
SQL> CREATE TABLE COLDEPT AS SELECT * FROM DEPT WHERE 1=0;
```

SYNTAX3: A NEW TABLE WITH COPY OF SPECIFIC COLUMNS FROM OLD TABLE:

==
CREATE TABLE <NEW TABLE NAME> AS SELECT <SPECIFIC COLUMN NAMES> FROM <OLD TABLE NAME>;

EX:
SQL> CREATE TABLE SPECCOL AS SELECT EMPNO,ENAME,SAL FROM EMP;

SYNTAX3: A NEW TABLE WITH COPY OF SPECIFIC ROWS FROM OLD TABLE:
==
CREATE TABLE <NEW TABLE NAME> AS SELECT * FROM <OLD TABLE NAME> WHERE <DATA CONDITION>;

EX:
SQL> CREATE TABLE SPECROWS AS SELECT * FROM EMP WHERE DEPTNO=20;

HOW TO COPY DATA FROM ONE TABLE TO ANOTHER TABLE:
===
SYNTAX:
========
INSERT INTO <DESTINATION TABLE NAME> SELECT * FROM <SOURCE TABLE NAME>;

BASIC RULES:
=============

1. NO.OF COLUMNS SHOULD BE SAME IN BOTH TABLES.
2. DATATYPES OF COLUMNS MUST BE SAME IN BOTH TABLES

EX:
SQL> INSERT INTO DDEPT SELECT * FROM DEPT;

Insert all:
========

- to insert multiple rows data into multiple tables at a time. but those rows data from an existing table only.

syntax:
=======
```
INSERT ALL INTO <TN1> VALUES(<COL1>,<COL2>,......)
INTO <TN2> VALUES(<COL1>,<COL2>,.........................)
INTO <TN3> VALUES(<COL1>,<COL2>,.........................)
.........................................................................................
.........................................................................................
INTO <TN n> VALUES(<COL1>,<COL2>,.........................) SELECT * FROM <OLD TABLE
NAME>;
```

EX:
```
SQL> CREATE TABLE TEST1 AS SELECT * FROM DEPT WHERE 1=0;

Table created.

SQL> CREATE TABLE TEST2 AS SELECT * FROM DEPT WHERE 1=0;

Table created.

SQL> CREATE TABLE TEST3 AS SELECT * FROM DEPT WHERE 1=0;

SQL> INSERT ALL INTO TEST1 VALUES(DEPTNO,DNAME,LOC)
  2  INTO TEST2 VALUES(DEPTNO,DNAME,LOC)
  3  INTO TEST3 VALUES(DEPTNO,DNAME,LOC)
  4  SELECT * FROM DEPT;
```

ALIAS NAMES:
=============

- temporary name to columns and also table.

    I) column level:
    =============

        - when we create alias name to column.

    II) table level
    ===========

        - when we create alias name to table.

syntax:
=======

select <column name1> <column alias name1>,<column name2> <column alias name2>,.....from <tn> <table alias name>;

Ex:
```
SQL> SELECT DEPTNO X,DNAME Y,LOC Z FROM DEPT D;
```

CONCATENATION OPERATOR(||):
============================

- TO ADD TWO STRING EXPRESSIONS.

SYNTAX:
=======

    <STRING1> || <STRING2>

EX:

```
SQL> SELECT 'WELCOME'||' '||'TO'||' '||'ORACLE' FROM DUAL;

'WELCOME'||''||'T
-----------------
WELCOME TO ORACLE
```

EX:

```
SQL> SELECT 'Mr.'||ENAME||' '||'IS WORKING AS A'||' '||JOB FROM EMP;

'MR.'||ENAME||''||'ISWORKINGASA'||''||J
---------------------------------------
Mr.SMITH IS WORKING AS A CLERK
```

DISTINCT KEYWORD:
==================

- to eliminate duplicate values from a column.

syntax:
=======

distinct column name

EX:

```
SQL> SELECT DISTINCT JOB FROM EMP;
SQL> SELECT DISTINCT DEPTNO  FROM EMP;
```

1) PAGESIZE n:
=============

- NO.OF ROWS DISPLAY PER A PAGE.
- "n" --- NO.OF ROWS
- BY DEFAULT A PAGE IS DISPLAY 14 ROWS.

SYNTAX:
========

```
SET PAGESIZE n;
```

EX:

```
SET PAGESIZE 100;
```

2) LINES n:

=========

- BY DEFAULT EACH LINE IS HAVING 80 BYTES.
- "n" NO.OF BYTES

SYNTAX:
========
SET LINES n;

EX:
SET LINES 100;

100 CHAR'S
--(80 BYTES) 100C / 100CHARS(1C=1B)

100 CHAR'S
--(80 BYTES) 100C / 100CHARS(1C=1B)

100 CHAR'S
--(80 BYTES) 100C / 100CHARS(1C=1B)

100 CHAR'S
--(80 BYTES) 100C / 100CHARS(1C=1B)

14 ROWS ---- DATA ----- DISPLAY ---14 LINES

ORACLE(SQL & PL/SQL) @ 7:30 PM (IST) by Mr.Sudhakar.L
Day-1 https://youtu.be/c2JaUN2lO_I
Day-2 https://youtu.be/WI3nfToIVI0
Day-3 https://youtu.be/RthH6ugYp1Q
Day-4 https://youtu.be/aAgsi_OI090
Day-5 https://youtu.be/nrfU9ofKpBk
Day-6 https://youtu.be/r30xw_cHj1Q