- Linesize and pagesize
- Data types
- Different way of Insertion
- Delete column
- Add column
- Update datatype of column
- set column values

Linesize and pagesize

```
SQL> create table table7(data1 varchar2(50), data2 varchar(50));

Table created.

SQL> insert into table7(data1,data2) values ('ramesh','kumar');

1 row created.

SQL> select * from table7;

DATA1
--------------------------------------------------
DATA2
--------------------------------------------------
ramesh
kumar
```

Set linesize value

```
SQL> set linesize 150
SQL> select * from table7;

DATA1                                              DATA2
-------------------------------------------------- --------------------------------------------------
ramesh                                             kumar
```

```
SQL> select * from table7;

DATA1                                              DATA2
-------------------------------------------------- --------------------
ramesh                                             kumar
ramesh1                                            kumar
suresh                                             kumar
ramesh                                             kuma2
ramesh                                             kum3
ramesh                                             kum4
ramesh                                             kum7
ramesh                                             kum8
ramesh                                             kum9
ramesh                                             kuma10
ramesh                                             kuma11

DATA1                                              DATA2
-------------------------------------------------- --------
ramesh                                             kuma12
ramesh                                             kuma14
ramesh                                             kuma17
```

Set pazesize value

```
SQL> set pagesize 17
SQL> select * from table7;

DATA1                                              DATA2
-------------------------------------------------- --------------------
ramesh                                             kumar
ramesh1                                            kumar
suresh                                             kumar
ramesh                                             kuma2
ramesh                                             kum3
ramesh                                             kum4
ramesh                                             kum7
ramesh                                             kum8
ramesh                                             kum9
ramesh                                             kuma10
ramesh                                             kuma11
ramesh                                             kuma12
ramesh                                             kuma14
ramesh                                             kuma17

14 rows selected.
```

Value=0 ?

Multiple Insertion

DUAL :

- It is a table that is automatically created by Oracle Database along with the data dictionary.

- DUAL is in the schema of the user SYS but is accessible by the name DUAL to all users.

```
SQL> SELECT 2+2;
SELECT 2+2
         *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected


SQL> SELECT 2+2
  2  FROM DUAL;
         4
```

It has one column, DUMMY, defined to be VARCHAR2(1), and contains one row with a value X.

There may be a situation where we want to query something that is not from a table.

For example, getting the current date or querying a simple arithmetic expression like 2+2. In Oracle, clause FROM is not exceptional. If we don't write the FROM clause in Oracle, we'll get an error

1.
```
Insert into table_name values(value1,value2);
Insert into table_name values(value1,value2);
Insert into table_name values(value1,value2);
```

```
SQL> insert into table7 values('subin','kumar');

1 row created.

SQL> insert into table7 values('ramesh','kumar');

1 row created.
```

2.
```
Insert into table_name (col1,col2)
Select value1,value2 from dual
Union all
Select value1,value2 from dual
Union all
Select value1,value2 from dual
```

```
SQL> insert into table7(data1,data2)
  2  select 'Shree','Prakash' from dual
  3  union all
  4  select 'pritam','kumar' from dual
  5  ;

2 rows created.
```

3.
```
Insert all
Insert into table_name (col1,col2) values(value1,value2)
Insert into table_name (col1,col2) values(value1,value2)
                                ...
Select 1 from dual;
```

```
SQL> insert all
  2  into table7(data1,data2)values('Dinesh','rawat')
  3   into table7(data1,data2)values('Dinesh','rawat')
  4  select 1 from dual;

2 rows created.
```

4. Use SQL*Loader to load csv. file

Character data type

CHAR : char_name CHAR(length)

- fixed-length data type
- once initialized cannot change the size at execution time.
- Static datatype.
- store normal characters and alphanumeric characters too.
- maximum length of 2000 bytes of characters
- If you store 5 characters in char(10), then the 5 bytes will be stored by oracle and the remaining 5 bytes will be padded to the right side leading to memory wastage as shown in the example in the latter part of the article.

VARCHAR : char_name VARCHAR(length)

- variable-length data type
- can change the size of the character at the time of the execution.
- Dynamic datatype.
- store normal characters and alphanumeric characters too.
- maximum length of **4000 bytes**.
- Also, for every one character, one byte is stored in the memory.
- VARCHAR is an ANSI Standard that is used to distinguish between Null and Empty Strings

VARCHAR2 :

char_name VARCHAR2(length)

- same as VARCHAR in the oracle database.
- The main difference is that VARCHAR is ANSI Standard and VARCHAR2 is Oracle standard.

.

ANSI-SQL: NULL is a specific value or mark that is used to indicate the absence of any data value.

NULL in Oracle

- Unassigned value
- Unknown value
- Each NULL is a unique value
  - NULL !=NULL
- Arithmetic operation cant be performed
  - If performed return value is NULL
- Null is untyped in Oracle

```
SQL> create table person(name varchar2(20), age number);

Table created.

SQL> insert into person values('ramesh',3);

1 row created.

SQL> insert into person values(null,4);

1 row created.

SQL> insert into person values('suresh',null);

1 row created.

SQL> select * from person;

NAME                          AGE
-------------------- ----------
ramesh                         3
                               4
suresh
```

create table comparision(d1 char(10),d2 varchar(10),d3 varchar2(10));

insert into comparision(d1,d2,d3)values('rahul','rahul','rahul');

```
SQL> select * from comparision;

D1         D2         D3
---------- ---------- ----------
rahul      rahul      rahul

SQL> select dump(d1) from comparision;

DUMP(D1)
--------------------------------------------------
Typ=96 Len=10: 114,97,104,117,108,32,32,32,32,32

SQL>  select dump(d2) from comparision;

DUMP(D2)
--------------------------------------------------
Typ=1 Len=5: 114,97,104,117,108

SQL> select dump(d3) from comparision;

DUMP(D3)
--------------------------------------------------
Typ=1 Len=5: 114,97,104,117,108
```

Return a varchar2 value that contains the datatype code, the length in bytes, and the internal representation of the expression

Decimal representation

VARCHAR2	4000 bytes	1
NUMBER	21 bytes	2
LONG	2^31-1 bytes	8
ROWID	10 bytes	11
DATE	7 bytes	12

NVARCHAR2(size): Variable-length Unicode character

LONG: Character data of variable length up to 2 gigabytes

Large Objects (LOBs) are a set of data types that are designed to hold large amounts of data. A LOB can hold up to a maximum size ranging from 8 terabytes to 128 terabytes

Large Objects (LOBs) hold large amounts of data.
maximum size ranging from 8 terabytes to 128 terabytes

BLOB	**Binary Large Object** **Stores any kind of data in binary format.** **Typically used for multimedia data such as images, audio, and video.**
CLOB	**Character Large Object** Stores string data in the database character set format. Used for large strings or documents that use the database character set exclusively. Characters in the database character set are in a fixed width format.
NCLOB	**National Character Set Large Object** Stores string data in National Character Set format. Used for large strings or documents in the National Character Set. Supports characters of varying width format.
BFILE	**External Binary File** A binary file stored outside of the database in the host operating system file system, but accessible from database tables. BFILEs can be accessed from your application on a **read-only** basis. Use BFILEs to store static data, such as image data, that is not manipulated in applications.

NUMBER data type

store numeric values that can be negative or positive

Number(Precision, scale)

- The precision
  - is the number of digits in a number.
  - It ranges from 1 to 38.
- The scale is the number of digits to the right of the decimal point in a number.
  - It ranges from -84 to 127.

- 1234.56 has a precision of 6 and a scale of 2

  NUMBER(6,2).
- Both precision and scale are in decimal digits and optional.
- skip the precision and scale, Oracle uses the maximum range and precision for the number

- Store numeric values with the maximum range and precision: **NUMBER**
- The following syntax defines a fixed-point number: **NUMBER(p,s)**
- To define an integer, you use the following form: **NUMBER(p)**
- Fixed-point number with precision p and scale of zero: **NUMBER(p,0)**
- Oracle allows the scale to be negative, for example the following number will round the numeric value to hundreds. **NUMBER(5,-2)**

```
SQL> create table ndate(data1 number, data2 number(3), data3 number(3,2),
  2  data4 number(5,2),data5 number(6,1),data6 number(6,-2), data7
  3  number(2,7));
```

CREATION

```
SQL> describe ndate;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DATA1                                              NUMBER
 DATA2                                              NUMBER(3)
 DATA3                                              NUMBER(3,2)
 DATA4                                              NUMBER(5,2)
 DATA5                                              NUMBER(6,1)
 DATA6                                              NUMBER(6,-2)
 DATA7                                              NUMBER(2,7)
```

Property

```
SQL> insert into ndate(data1,data2,data3,data4,data5,data6,data7)values
  2  (248.79,248.79,1.12, 248.79,248.79, 248.79,.0000123);
```

INSERTION

```
1 row created.
```

DISPLAY

```
SQL> select * from ndate;

     DATA1      DATA2      DATA3      DATA4      DATA5      DATA6      DATA7
---------- ---------- ---------- ---------- ---------- ---------- ----------
    248.79        249       1.12     248.79      248.8        200    .0000012
```

```
SQL> insert into ndate(data1,data2,data3,data4,data5,data6,data7)values
  2  (248.79,248.79,248.79,248.79,248.79, 248.79,.00000123);
(248.79,248.79,248.79,248.79,248.79, 248.79,.00000123)
                 *
ERROR at line 2:
ORA-01438: value larger than specified precision allowed for this column
```

Float

- subtype of number
- precision may or may not be specified
- scale cannot be specified
- It is interpreted from data

---

create table diff(data1 number(5,2), data2 float(5));

insert into diff(data1, data2)values(6.89,6.89);

insert into diff(data1, data2)values(1.34,1.34);

insert into diff(data1, data2)values(126.45,126.45);

insert into diff(data1, data2)values(500.34,12345);

insert into diff(data1, data2)values(1234.42,12345);

What will happen?

SQL> **select * from diff;**

DATA1	DATA2
6.89	6.9
1.34	1.3
126.45	130
500.34	12000

BINARY_FLOAT	32-bit floating point number.	This data type requires 4 bytes.
BINARY_DOUBLE	64-bit floating point number.	This data type requires 8 bytes.
ROWID	The unique address (base 64 string representing) of a row in its table	

```
SQL> create table table2(data1 binary_float, data2 binary_double);

Table created.

SQL> insert into table2(data1,data2)values(23456,23456);

1 row created.

SQL> select * from table2;

     DATA1      DATA2
---------- ----------
2.346E+004 2.346E+004

SQL> select rowid from table2;

ROWID
------------------
AAAM1TAAEAAAAIlAAA
```

Date

create table dates(doj date);

insert into dates(doj)values('13-nov-2022');

alter session set nls_date_format='DD-mm-yyyy hh24:mi:ss';

SQL> select * from dates;

DOJ

13-11-2022 00:00:00

```
SQL> insert into dates(doj)values(TO_DATE('2003/05/03 21:02:44', 'yyyy/mm/dd hh24:mi:ss'));

1 row created.

SQL> insert into dates(doj)values(TO_DATE('01/01/2023', 'dd/mm/yyyy'));

1 row created.

SQL> insert into dates(doj)values(TO_DATE('01/05/2023','mm/dd/yyyy'));

1 row created.

SQL> insert into dates(doj)values(TO_DATE('01/05/23','mm/dd/yy'));

1 row created.

SQL> insert into dates(doj)values(TO_DATE('01-05-1999','dd-mm-yyyy'));

1 row created.
```

```
SQL> select * from dates;

DOJ
---------
03-MAY-03
01-JAN-23
05-JAN-23
05-JAN-23
01-MAY-99
```

```
SQL> select * from dates;

DOJ
--------------------
21:02:44 03-05-2003
00:00:00 01-01-2023
00:00:00 05-01-2023
00:00:00 05-01-2023
00:00:00 01-05-1999
```

```
SQL> select * from dates;

DOJ
--------------------
03-05-2003 21:02:44
01-01-2023 00:00:00
05-01-2023 00:00:00
05-01-2023 00:00:00
01-05-1999 00:00:00
```

ADD column

ALTER TABLE table_name

ADD (column_name_1 data_type constraint,
column_name_2 data_type constraint, ...);

```
SQL> alter table dates add(name varchar2(50), age number);

Table altered.
```

```
SQL> desc dates;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DOJ                                                DATE
 NAME                                               VARCHAR2(50)
 AGE                                                NUMBER
```

Dropping Columns

alter table table_name drop unused columns;

```
SQL> alter table dates set unused (age);

Table altered.

SQL> desc dates;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DOJ                                                DATE
 NAME                                               VARCHAR2(50)
```

Logical delete

alter table table_name drop unused columns; **Physical delete**

alter table table_name drop (column_name1, column_name2);

Direct

Physical delete

```
SQL> alter table dates drop (name);

Table altered.

SQL> desc dates;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 DOJ                                                DATE
```

Modify

ALTER TABLE table_name MODIFY (column_name_1 action, column_name_2 action, ...);

alter table dates modify

(roll_no varchar(20));

```
SQL> alter table dates add(roll_no number);

Table altered.

SQL> select * from dates;

DOJ                       ROLL_NO
------------------- ----------
21:02:44 03-05-2003
00:00:00 01-01-2023
00:00:00 05-01-2023
00:00:00 05-01-2023
00:00:00 01-05-1999

SQL> alter table dates modify(roll_no varchar(20));

Table altered.

SQL> desc dates;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------
 DOJ                                                DATE
 ROLL_NO                                            VARCHAR2(20)
```

Update column value

```
UPDATE
   table_name
SET
   column1 = value1,
   column2 = value2,
   column3 = value3,
   ...
WHERE
   condition;
```

update dates set roll_no='coe19d002';

```
SQL> select * from dates;

DOJ                  ROLL_NO
-------------------- --------------------
00:00:00 05-01-2023

SQL> update dates set roll_no='coe19d002';

1 row updated.

SQL> select * from dates;

DOJ                  ROLL_NO
-------------------- --------------------
00:00:00 05-01-2023 coe19d002

SQL> insert into dates(doj)values(TO_DATE('01/06/2023','mm/dd/yy'));

1 row created.

SQL> update dates set roll_no='coe19d005' where roll_no='coe19d002';

1 row updated.

SQL> select * from dates;

DOJ                  ROLL_NO
-------------------- --------------------
00:00:00 05-01-2023 coe19d005
00:00:00 06-01-2023
```

```
SQL> update dates set roll_no='coe19doo2' where doj='00:00:00 06-01-2023';

1 row updated.

SQL> select * from dates;

DOJ                  ROLL_NO
-------------------- --------------------
00:00:00 05-01-2023 coe19d005
00:00:00 06-01-2023 coe19doo2
```