PARTITION TABLE:

GENERALLY, PARTITIONS ARE CREATED ON VERY LARGE-SCALE DATABASE TABLES FOR DIVIDING INTO MULTIPLE SMALL PARTS AND EACH PART IS CALLED AS "PARTITION".

- BY SPLITTING A LARGE TABLE INTO SMALLER PARTS THEN DATA CAN ACCESS VERY FAST BECAUSE THERE IS LESS DATA TO SACN INSTEAD OF LARGE DATA OF A TABLE.

TYPES OF PARTITIONS:

1) RANGE PARTITION

2) LIST PARTITION

3) HASH PARTITION

- IF WE WANT TO ACCESS A PARTICULAR PARTITION THEN WE FOLLOW THE FOLLOWING,

SYNTAX:

```
SQL> SELECT * FROM <TN> PARTITION (<PARTITION NAME>);
```

1) RANGE PARTITION:

- IN THIS METHOD WE ARE CREATING PARTITIONS TABLE BASED ON A PARTICULAR RANGE VALUE.

SYNTAX:

```
CREATE TABLE <TN> (<COLUMN NAME1> <DATATYPES>[SIZE], .....................................) PARTITION BY RANGE (<KEY COLUMN NAME>) (PARTITION <PARTITION NAME1> VALUES LESS THAN(VALUE), PARTITION <PARTITION NAME2> VALUES LESS THAN(VALUE), .........................................................);
```

EX:

```
CREATE TABLE TEST1(EID INT, ENAME VARCHAR2(10), SAL NUMBER (10)) PARTITION BY RANGE(SAL) (PARTITION P1 VALUES LESS THAN (1000),PARTITION P2 VALUES LESS THAN (2000), PARTITION P3 VALUES LESS THAN (3000));
```

TESTING:

```
SQL> INSERT INTO TEST1 VALUES(1,'SAI',2500);

SQL> INSERT INTO TEST1 VALUES(2,'JONES',500);

..........................................................................;

...........................................................................;
```

CALLING A PARTICULAR PARTITION:

```
SQL> SELECT * FROM TEST1 PARTITION(P1);


      EID          ENAME          SAL
-------------     ----------     ----------
       2           JONES          500
```

2) LIST PARTITION:

- IN THIS METHOD WE ARE CREATING PARTITIONS BASED ON LIST OF VALUES.

SYNTAX:

```
CREATE TABLE <TN> (<COLUMN NAME1> <DATATYPE>[SIZE],
.......................................)

PARTITION BY LIST (<KEY COLUMN NAME>) (PARTITION
<PARTITION NAME1> VALUES (VALUE1, VALUE2, ……),
PARTITION <PARTITION NAME2> VALUES (VALUE1, VALUE2,
……), …………, PARTITION OTHERS VALUES(DEFAULT));
```

EX:

```
CREATE TABLE TEST2(SNO INT, CNAME VARCHAR2(10))

PARTITION BY LIST(CNAME) (PARTITION P1
VALUES('ORACLE','MYSQL'),

PARTITION P2 VALUES('JAVA','PHP'), PARTITION OTHERS
VALUES(DEFAULT));
```

TESTING:

```
SQL> INSERT INTO TEST2 VALUES(1,'ORACLE');

SQL> INSERT INTO TEST2 VALUES(2,'C');

.................................................................
```

CALLING A PARTICULAR PARTITION:

```
SQL> SELECT * FROM TEST2 PARTITION(P1);


      SNO                  CNAME
--------------              ----------
        1                  ORACLE
```

3) HASH PARTITION:

- IN THIS METHOD PARTITIONS ARE CREATED BY THE SYSTEM BY DEFAULT.

SYNTAX:

```
CREATE TABLE <TN> (<COLUMN NAME1> <DATATYPE>[SIZE],
.......................................) PARTITION BY HASH (<KEY COLUMN
NAME>) PARTITIONS <NUMBER>;
```

EX:

```
SQL> CREATE TABLE TEST3(SNO INT, SAL NUMBER (10))
PARTITION BY HASH(SAL) PARTITIONS 5;
```

NOTE: IF WE WANT TO VIEW ALL PARTITIONS INFORMATION IN ORACLE DATABASE THEN WE USE "USER_TAB_PARTITIONS" DATA DICTIONARY.

EX:

```
SQL> DESC USER_TAB_PARTITIONS;

SQL> SELECT PARTITION_NAME FROM USER_TAB_PARTITIONS

        WHERE TABLE_NAME='TEST3';
```

ADDING A NEW PARTITION:

SYNTAX:

```
ALTER TABLE <TN> ADD PARTITION <PARTITION NAME> VALUES
LESS THAN(VALUE);
```

EX:

```
SQL> ALTER TABLE TEST1 ADD PARTITION P4 VALUES LESS THAN
(4000);
```

DROPPING A PARTITION:

SYNTAX:

```
ALTER TABLE <TN> DROP PARTITION <PARTITION NAME>;
```

EX:

```
SQL> ALTER TABLE TEST1 DROP PARTITION P1;
```

NOTE: IF WE WANT TO KNOW WHETHER TABLE IS PARTITIONED OR NOT THEN WE USE "USER_TABLES" DATA DICTIONARY.

EX:

```
SQL> DESC USER_TABLES;
SQL> SELECT PARTITIONED FROM USER_TABLES WHERE
TABLE_NAME='EMP';
```