CHAPTER 5

STATEMENTS

Statements – they help us to create the table and insert the data.

There are 3 types of statements,

Auto commit / No Rollback

- structure → DDL – Data Definition Language – the various commands in DDL are :- Create, Drop, Truncate, Alter, Rename

Not-Auto Commit

- data → DML – Data Manipulation Language – the various commands in DML are :- Insert, Update, Delete
- TCL – Transaction Control Language – the various commands in TCL are :- Rollback, Commit, Savepoint

DCL → Grant Revoke to give permit to database

CREATE – It creates the table.

Before we study the Create command, let us first study the some of the basic datatypes we use in SQL.

& constraint → ~~Rule~~
NOT NULL
PK
Unique
check
FK

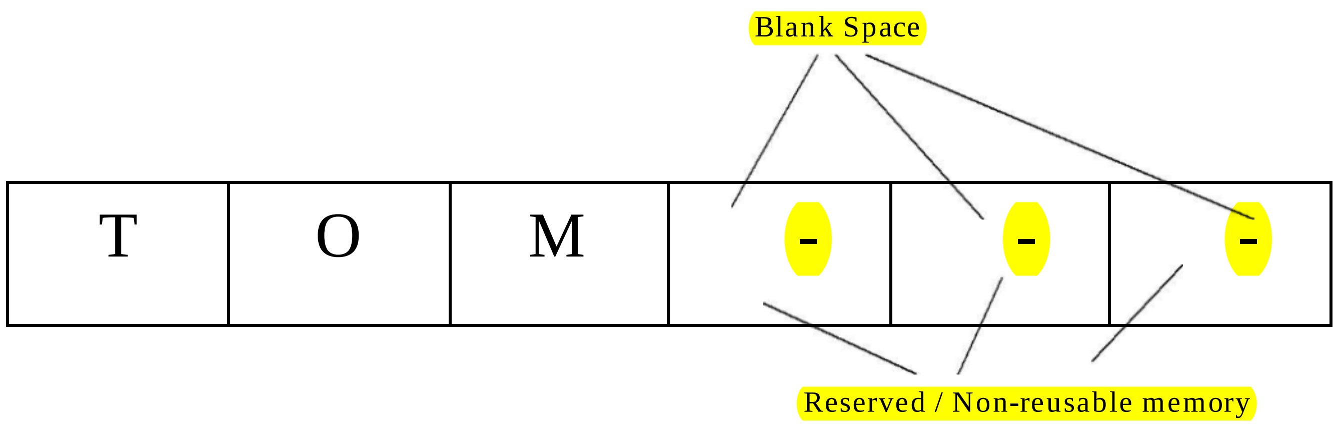
1) CHAR :-
It stores the fixed length character data.
It can store the alphanumeric data (i.e, numbers and characters).

2000 ch   oracle 10g → Now Varchar2 (4000 ch)

2) VARCHAR
It stores the variable length character data
It can store alphanumeric data.

Difference between CHAR & VARCHAR
Let us consider an example as shown below to explain the difference.

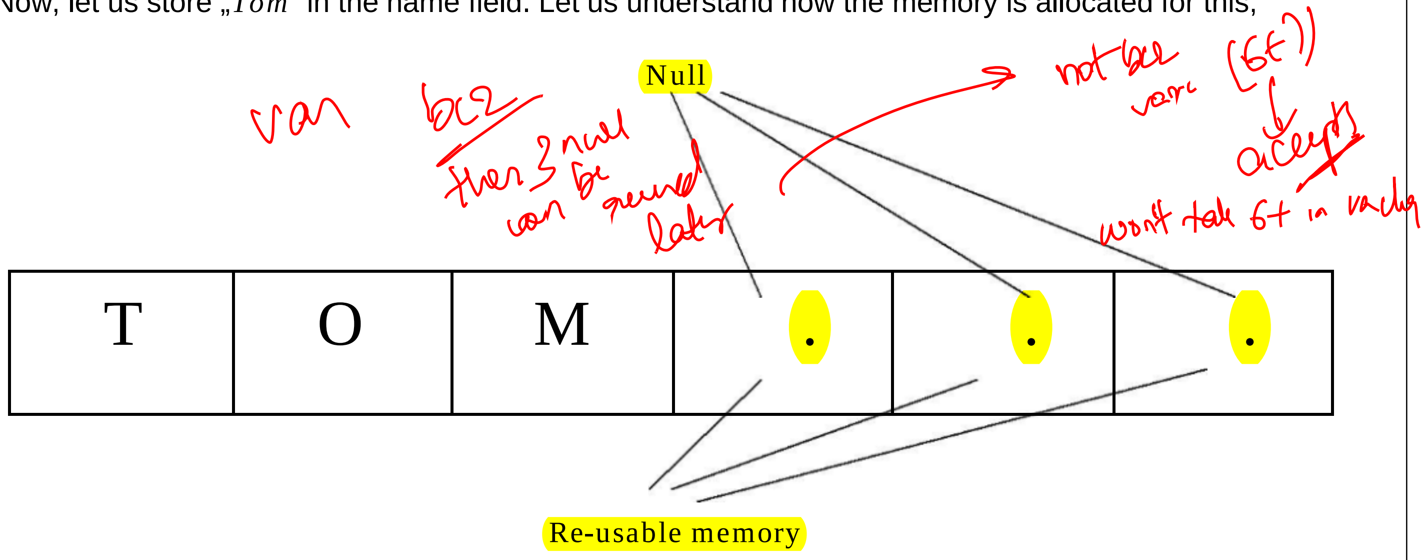
Name char (6) ;
Here we are defining name which is of 6characters in length.
Now, let us store „*Tom*" in the name field. Let us understand how the memory is allocated for this,



When we declare anything of type char, the memory is allocated as of the size given and its fixed length – hence it cannot be altered.

Now, when we give *tom*, it allocates 6 bytes for name char – only the 1st 3bytes are used to store Tom – the rest becomes waste as it is a blank space and it is reserved memory.
The length(name) = 6.

Name varchar (6) ;
Here we are defining name which is of 6characters in length.
Now, let us store „*Tom*‟ in the name field. Let us understand how the memory is allocated for this,



When we declare anything of type varchar, the memory is allocated as shown above and it is variable length

When we give *tom*, it allocates 6bytes for name varchar – only the 1st 3bytes are used to store tom – the remaining 3 fields becomes null. As we know the property of null – null does not occupy any memory space – thus the memory is not wasted here.
The length(name) = 3.

Another difference is : -
In char, maximum value we can store is 2000 characters
In varchar, maximum value we can store is 4000 characters.

3) NUMBER
- it stores numeric data.
For ex – 1) sal number(4) ;
Here the maximum possible value is 9999.

2) sal number (6, 2) ;
Here, 2 – scale (total number of decimal places)
6 – precision (total number of digits including decimal places) Maximum value is 9999.99
sal number (4, 3) ;
maximum value is 9.999
sal number (2, 2)
maximum value is .99

4) DATE
- it stores date and time
- no need to specify any length for this type.

For ex,   SQL > order_dt DATE;

Date is always displayed in the default format :-   dd – month – yy

> NOTE :-
> varchar2 – from 10g, varchar & varchar2 are the same.
> Earlier, varchar was supporting upto 2000 characters and varchar2 was supporting upto 4000 characters.

5) BLOB
Stands for – Binary Large Object
It stores binary data (images, movies, music files) within the database. It stores upto 4GB.

6) CLOB
Stands for – Character Large Object
It stores plain character data like varchar field upto 4GB.

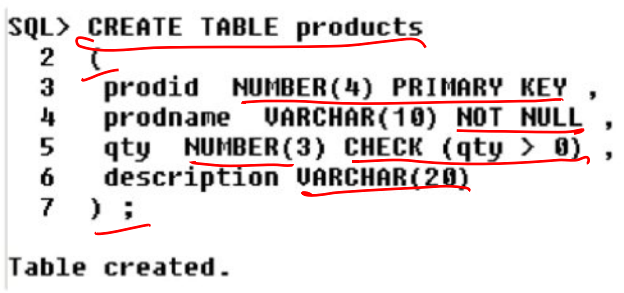
<u>Create the following tables</u>

PRODUCTS
ProdID (PK)
ProdName (Not Null)
Qty (Chk > 0)
Description

parent / master

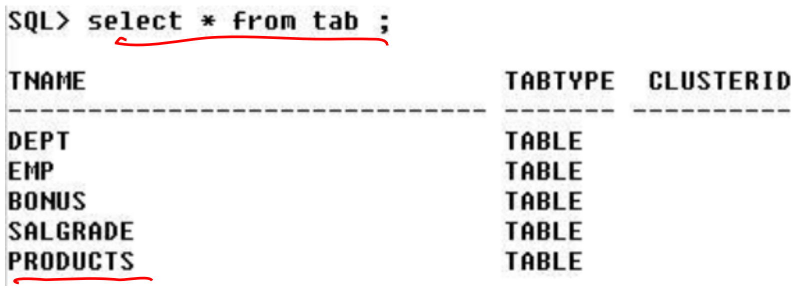
ORDERS
ProdID (FK from products)
OrderID (PK)
Qty_sold (chk > 0)
Price
Order_Date

FK refer → child / detail



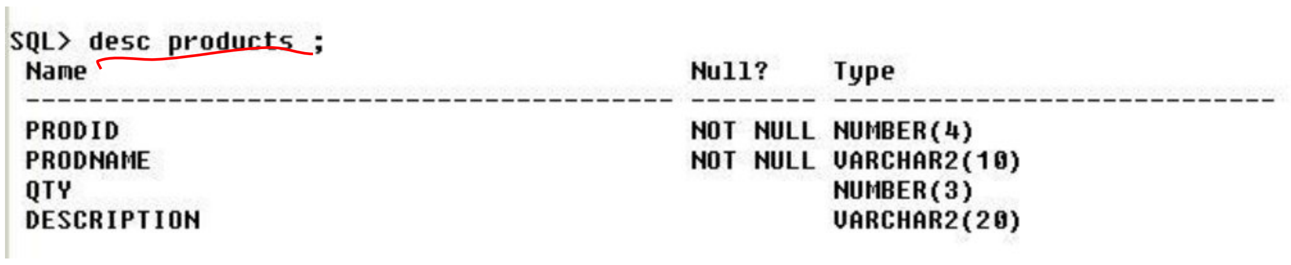
```
SQL> CREATE TABLE products
  (
   prodid  NUMBER(4) PRIMARY KEY ,
   prodname  VARCHAR(10) NOT NULL ,
   qty  NUMBER(3) CHECK (qty > 0) ,
   description VARCHAR(20)
  ) ;

Table created.
```

We can see that the table has been created.
Now, let us verify if the table has really been created and also the description of the table,



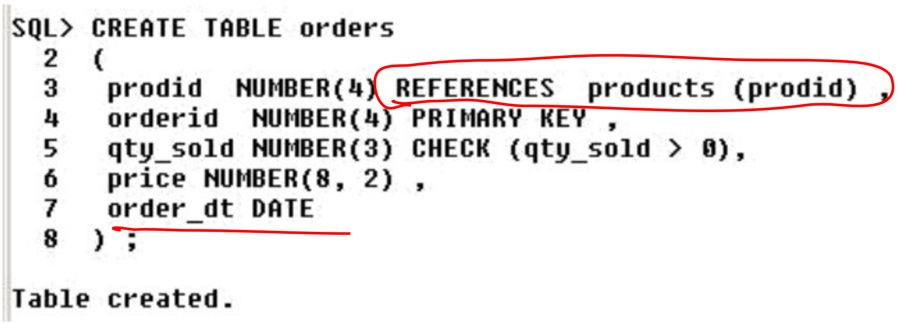
```
SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
```

The new table products has been added to the database.



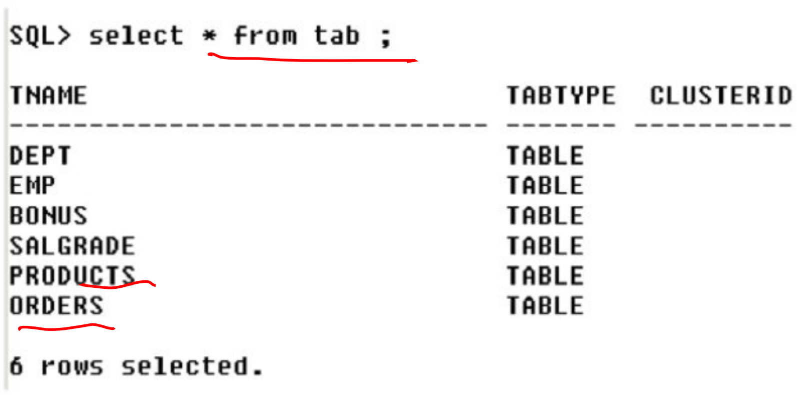
```
SQL> desc products ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                    NOT NULL NUMBER(4)
 PRODNAME                                  NOT NULL VARCHAR2(10)
 QTY                                                NUMBER(3)
 DESCRIPTION                                        VARCHAR2(20)
```

Thus, we get the description of the table products.



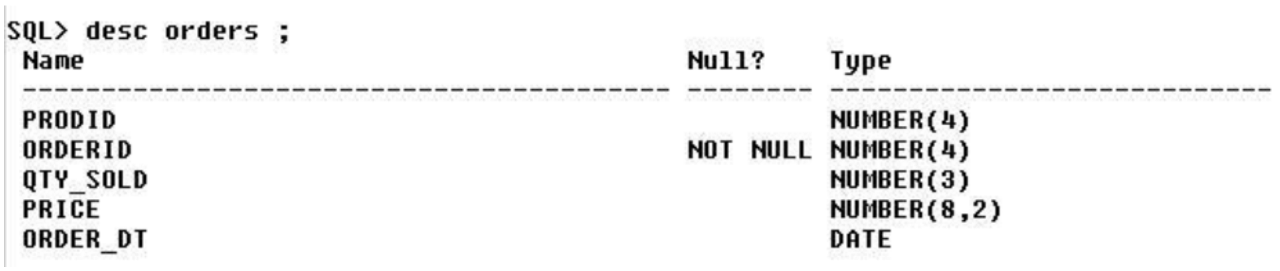
```
SQL> CREATE TABLE orders
  (
   prodid  NUMBER(4) REFERENCES  products (prodid) ,
   orderid  NUMBER(4) PRIMARY KEY ,
   qty_sold NUMBER(3) CHECK (qty_sold > 0),
   price NUMBER(8, 2) ,
   order_dt DATE
  ) ;

Table created.
```

The new table orders has been created. We can see from the above query how to reference a child table to the parent table using the references keyword.



```
SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
ORDERS                         TABLE

6 rows selected.
```

Thus we can verify that orders table has ben created and added to the database.

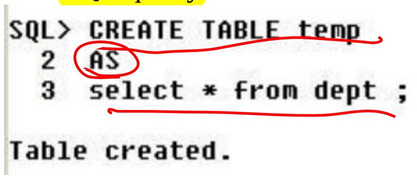


```
SQL> desc orders ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                             NUMBER(4)
 ORDERID                                   NOT NULL NUMBER(4)
 QTY_SOLD                                           NUMBER(3)
 PRICE                                              NUMBER(8,2)
 ORDER_DT                                           DATE
```

Thus, we get the description of the orders table.

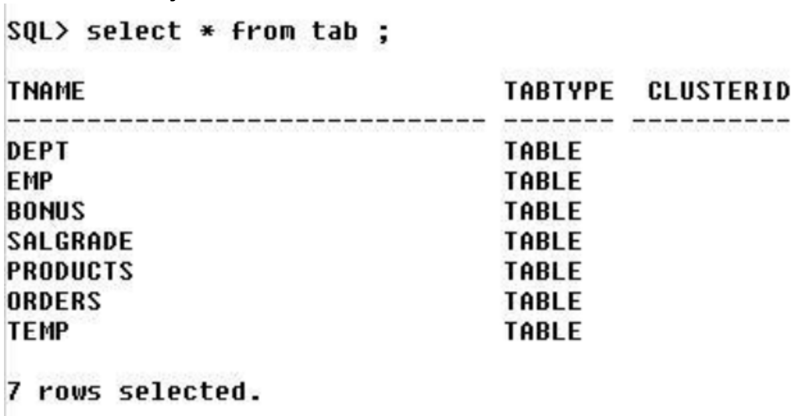
Creating a table from another table :-

Now, we will see how to create a table from another table – i.e, it duplicates all the records and the characterstics of another table.
The SQL query for it is as follows,



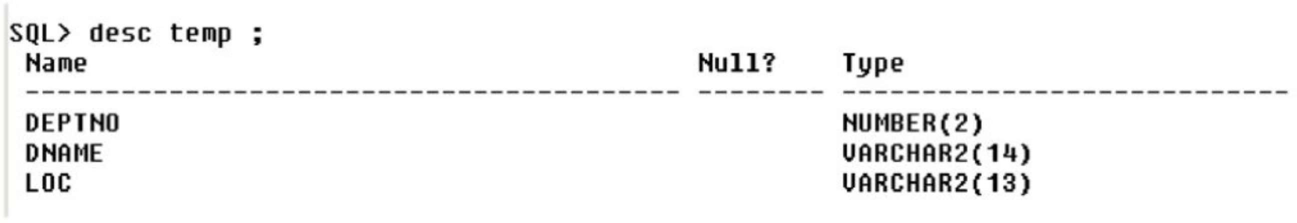
```
SQL> CREATE TABLE temp
  2  AS
  3  select * from dept ;

Table created.
```

Thus we can see that we have created another table temp from the table dept.
We can verify it as shown below,

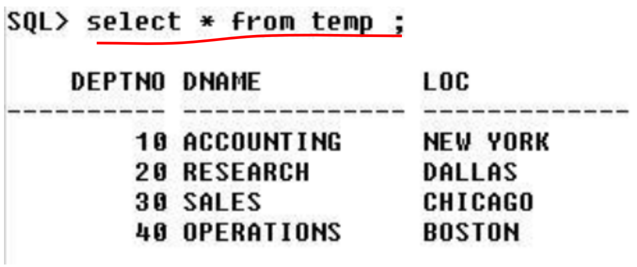


```
SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
ORDERS                         TABLE
TEMP                           TABLE

7 rows selected.
```

Thus, we can see that the table temp has been created.


```
SQL> desc temp ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

Thus, we can see that the table temp has copied the structure of the table dept. Here, we must observe that temp copies all the columns, rows and NOT NULL constraints only from the table dept. It never copies ~~PK, FK, Check~~ constraints.
Thus, when in the interview somebody asks you "I have a table which has about 1million records. How do I duplicate it into another table without using Insert keyword and without inserting it individually all the records into the duplicated table ?
Answer is - Use the above query of creating a table from another table and explain it.


```
SQL> select * from temp ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

Thus, from the above query – we can see that all the records of the table dept has been copied into the table temp.

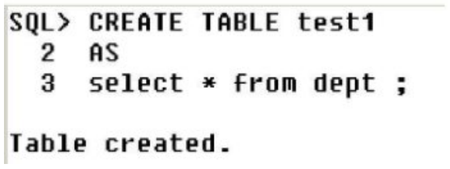
TRUNCATE

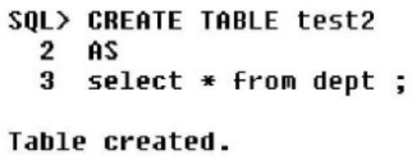
It removes all the data permanently, but the structure of the table remains as it is.
Ex – SQL > TRUNCATE TABLE test ;

DROP

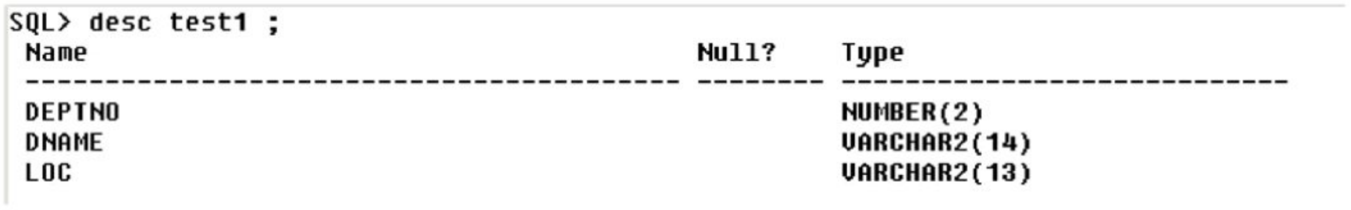
It removes both data and the structure of the table permanently from the database.
Ex – SQL > DROP TABLE test ;

Let us understand the difference between drop & truncate using the below shown example,

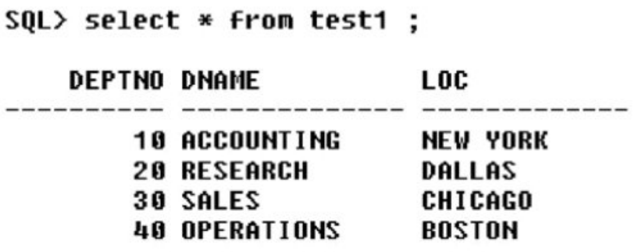

```
SQL> CREATE TABLE test1
  2  AS
  3  select * from dept ;

Table created.
```


```
SQL> CREATE TABLE test2
  2  AS
  3  select * from dept ;

Table created.
```

Let us create 2 tables Test1 and Test2 as shown above.

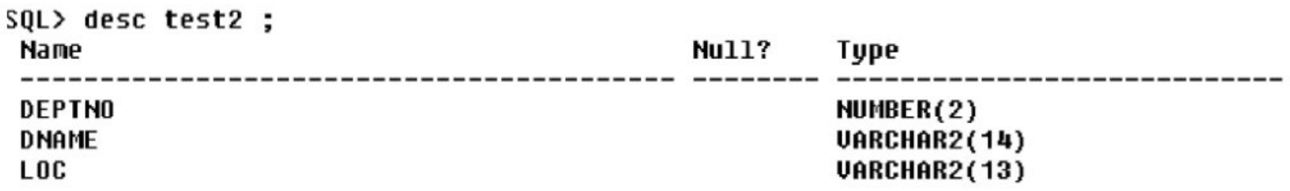


```
SQL> desc test1 ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

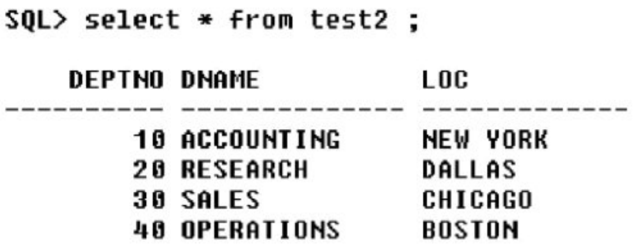


```
SQL> select * from test1 ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

The above shows the description of the table test1.



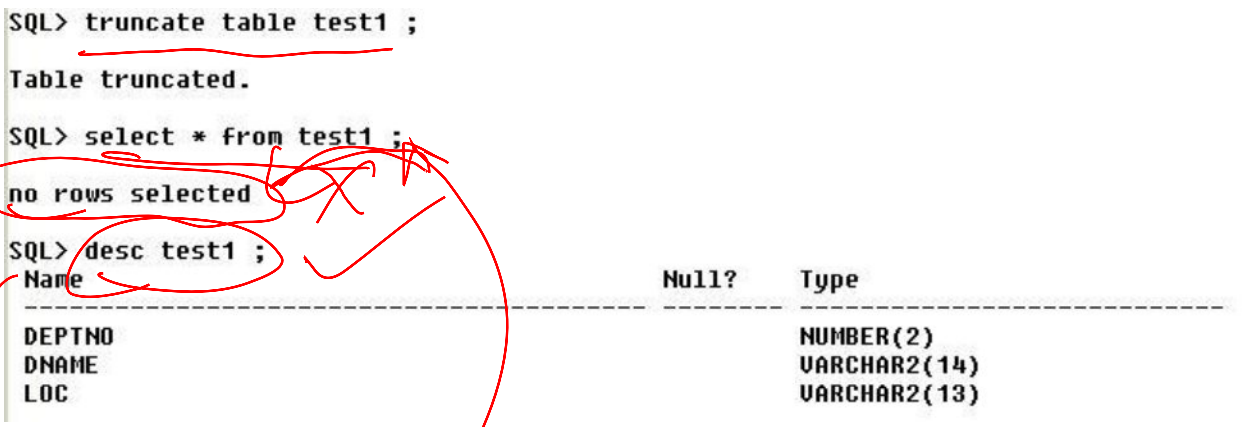
```
SQL> desc test2 ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```



```
SQL> select * from test2 ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

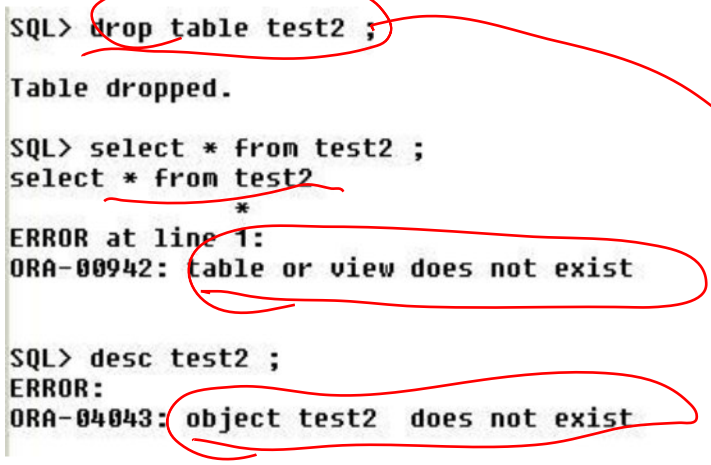
The above gives the description of the table Test2.

Now, let us use the Truncate query on Test1 and Drop query on Test2 and see the difference.



```
SQL> truncate table test1 ;

Table truncated.

SQL> select * from test1 ;

no rows selected

SQL> desc test1 ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

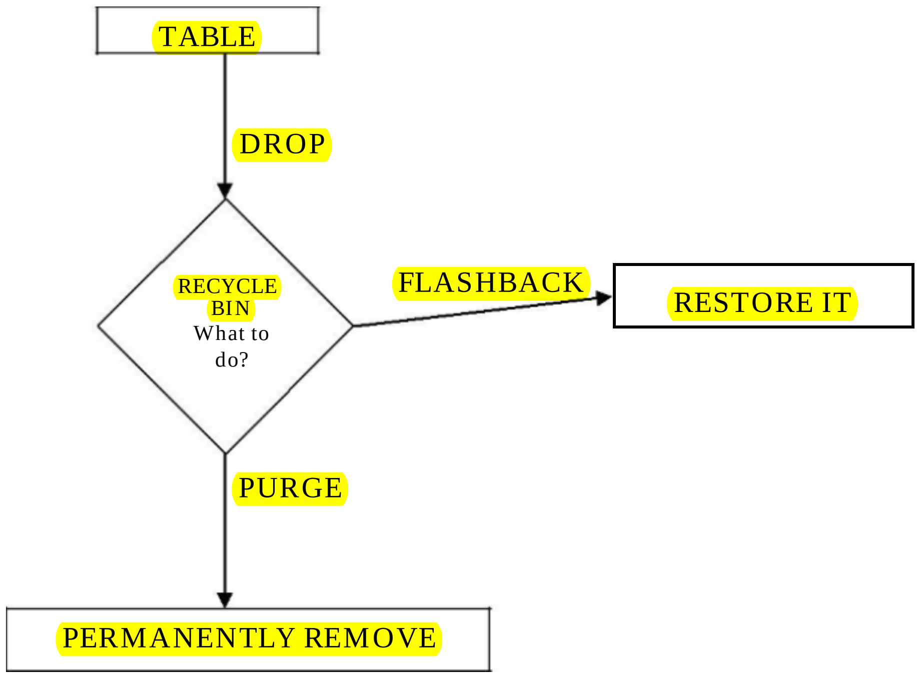
The above 3 queries show that – 1st query has the table test1 truncated.
2nd query – it shows no rows selected – thus only the records from the table has been removed. 3rd query – it shows that the structure of the table is still present. Only the records will be removed. Thus, this explains the truncate query.



```
SQL> drop table test2 ;

Table dropped.

SQL> select * from test2 ;
select * from test2
              *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> desc test2 ;
ERROR:
ORA-04043: object test2  does not exist
```

Thus from the above queries we can explain how drop works. 1st query – it drops the table. Thus – the entire structure and records of the table are dropped.
2nd and 3rd query – since, there is no table – select & desc query for test2 will throw an error. Thus, this explains the drop query.
Hence, we have seen the difference between drop & truncate query.

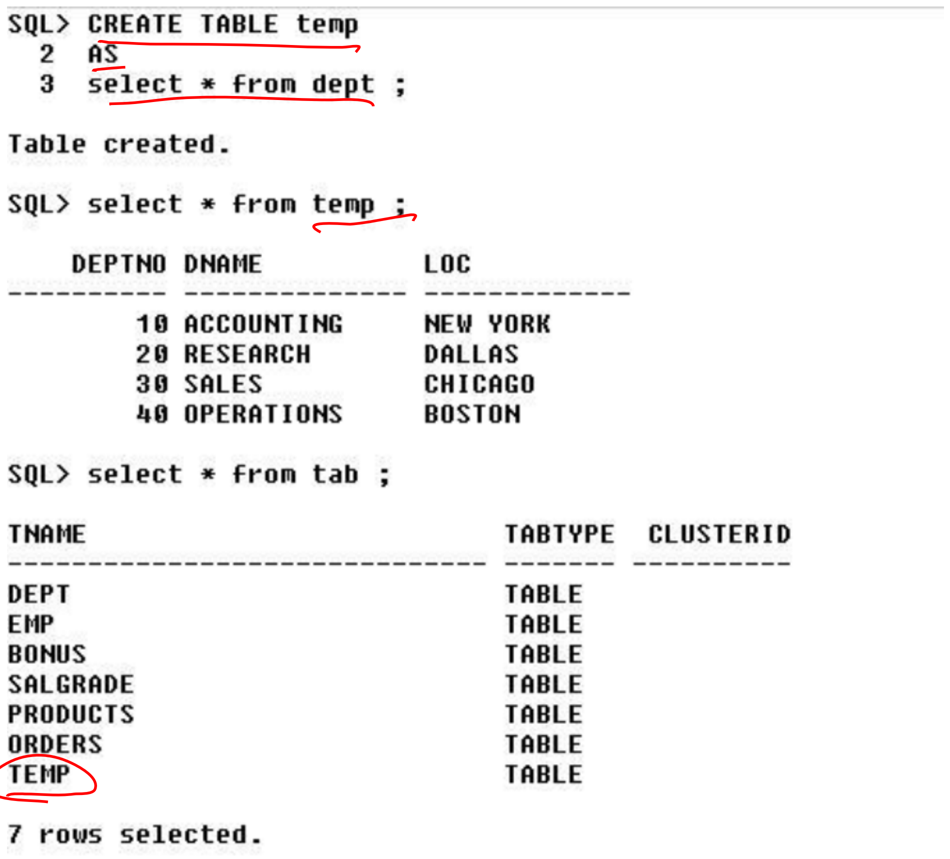
<u>10g Recycle Bin</u>



The functionality of Recycle Bin was introduced in Oracle 10G version only. Thus even though the table has been dropped, we can still restore it using flashback command or we can permanently remove it using the purge command.
This concept of Recycle bin was not there in the earlier versions of Oracle.

RENAME

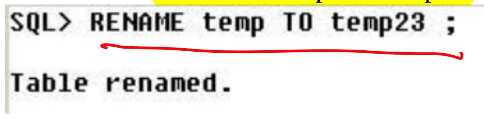
It renames a table.
For ex, let us see the query of how we do this renaming a table.



```
SQL> CREATE TABLE temp
  2  AS
  3  select * from dept ;

Table created.

SQL> select * from temp ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
ORDERS                         TABLE
TEMP                           TABLE

7 rows selected.
```

In the above 3queries – we have created a table temp which copies table dept – we see the records of the table temp – and also check if the table has really been created.

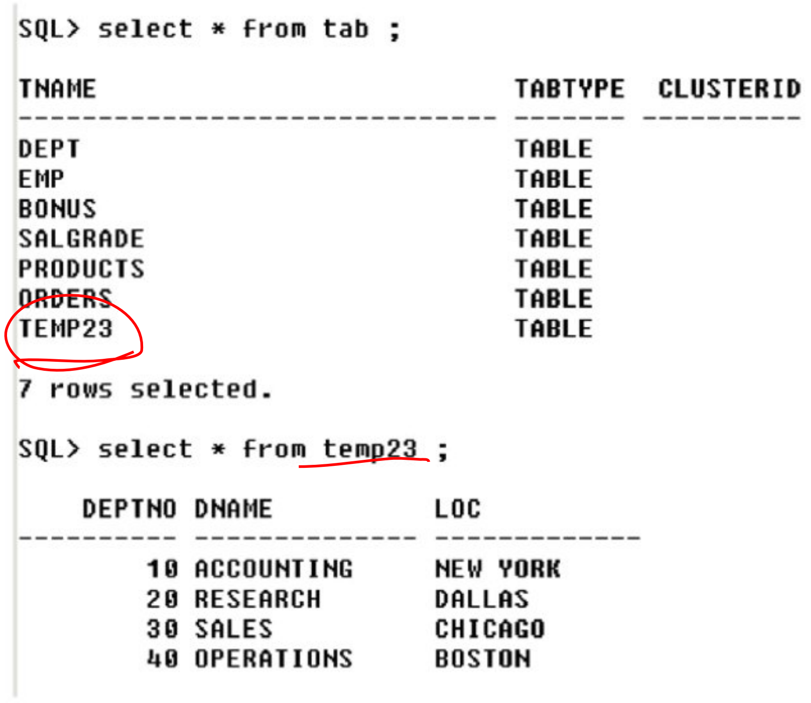
Now let us rename temp to temp23 as shown below,



```
SQL> RENAME temp TO temp23 ;

Table renamed.
```

The above query is used to rename a table.

Now let us verify the contents of the table and check if it has really been modified, See next page,



```
SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
ORDERS                         TABLE
TEMP23                         TABLE

7 rows selected.

SQL> select * from temp23 ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

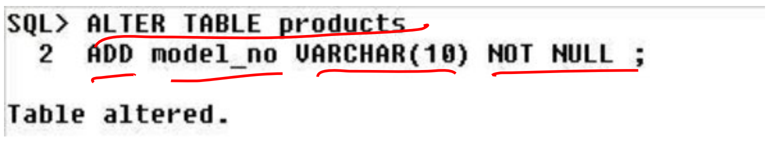
Thus the table has been renamed and its contents are verified.

ALTER

- this query alters / changes the structure of the table (i.e, - adding columns, removing columns, renaming columns etc).

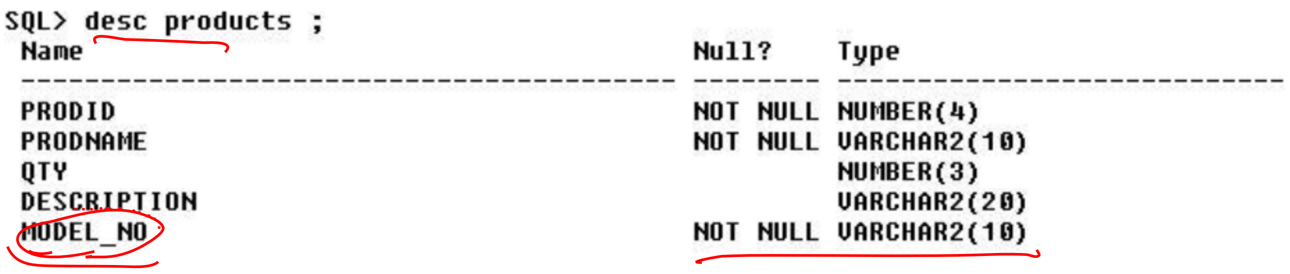
Now let us alter the table products (which we have created earlier).

1) Let us add a new column *‘model_no*‟ **to the** table.



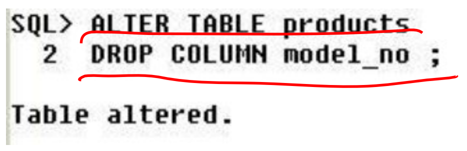
```
SQL> ALTER TABLE products
  2  ADD model_no VARCHAR(10) NOT NULL ;

Table altered.
```

Thus, a new column has been added. Lets verify it with the query shown below,



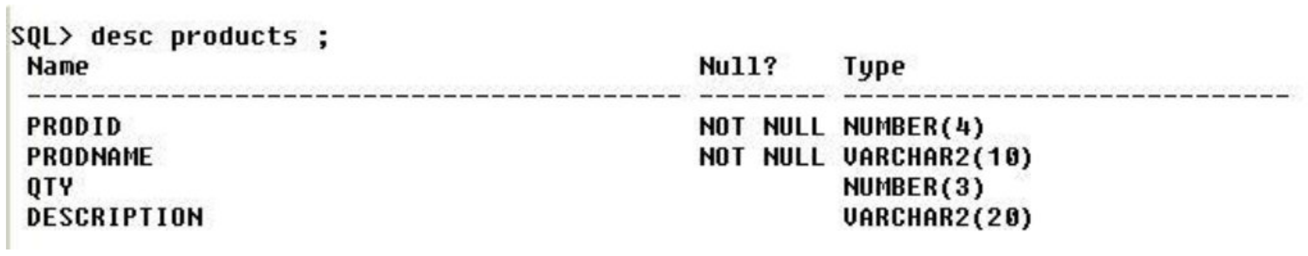
```
SQL> desc products ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                    NOT NULL NUMBER(4)
 PRODNAME                                  NOT NULL VARCHAR2(10)
 QTY                                                NUMBER(3)
 DESCRIPTION                                        VARCHAR2(20)
 MODEL_NO                                  NOT NULL VARCHAR2(10)
```

2) Now let us drop the column model_no from products.



```
SQL> ALTER TABLE products
  2  DROP COLUMN model_no ;

Table altered.
```

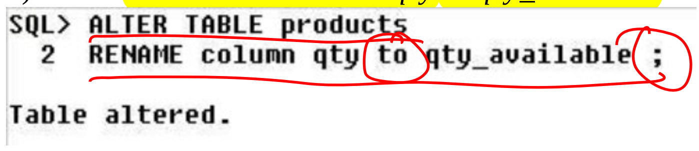
Thus, the column has been dropped.



```
SQL> desc products ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                    NOT NULL NUMBER(4)
 PRODNAME                                  NOT NULL VARCHAR2(10)
 QTY                                                NUMBER(3)
 DESCRIPTION                                        VARCHAR2(20)
```

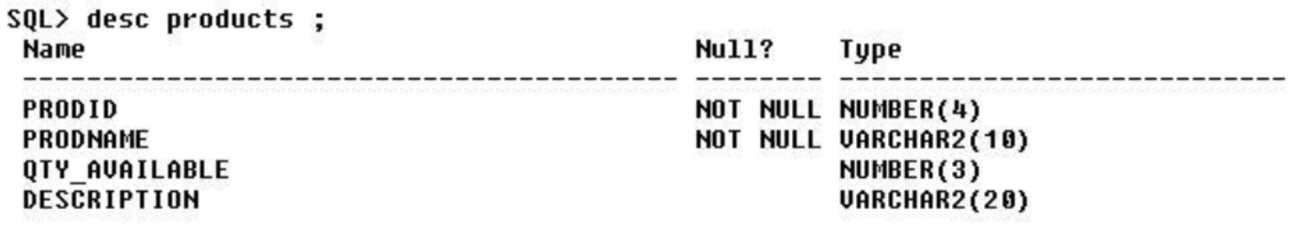
Thus, we can see from the description of the table – the column model_no has been dropped.

3) Let us rename the column *qty* to *qty_available*.



```
SQL> ALTER TABLE products
  2  RENAME column qty to qty_available ;

Table altered.
```

Let us verify if it has been renamed,



```
SQL> desc products ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                    NOT NULL NUMBER(4)
 PRODNAME                                  NOT NULL VARCHAR2(10)
 QTY_AVAILABLE                                      NUMBER(3)
 DESCRIPTION                                        VARCHAR2(20)
```

NOTE : *SELECT* is neither DML nor DDL. It does not belong to any group because it does not alter anything, it just displays the data as required by the user.