**Table**

**Create Table: emp**

* create table emp (eno number(5),ename varchar2(6),Sal number(5),eloc varchar2(6));

**To avoid duplicate entries create table with constraints:**

**Constraints:**

* constraints are used to specify rules for the data in a table.
* Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.
* Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.
* The following constraints are commonly used in SQL:
* **NOT NULL** - Ensures that a column cannot have a NULL value
* **UNIQUE** - Ensures that all values in a column are different
* **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* **FOREIGN KEY** - Prevents actions that would destroy links between tables
* **CHECK** - Ensures that the values in a column satisfies a specific condition
* **DEFAULT** - Sets a default value for a column if no value is specified
* **CREATE INDEX** - Used to create and retrieve data from the database very quickly.

1. **UNIQUE** = it won’t allow duplicates

2. **NOT NULL**. =.it won’t allow empty rows

3. **PRIMARY KEY** = unique + not null

4. **CHECK**

**Table creation with** **PRIMARY KEY:**

* Create table t1 (no number(5)primary key,ename varchar2(10),eloc varchar2(10));

**Insert data into table:**

* insert into emp values (1, ‘mouli’, 20000, sysdate, ‘hyd’);
* commit [ to save]
* roll back

**Update a table:**

* update emp set Sal= 30000 where eno=1;
* after update ‘commite’ to save changes

**To delete information in table:**

* delete from emp where eno=1;
* after delete we have to ‘commit’
* if we don’t want to delete Rollback before commit.

**To add new column to the table:**

* alter table emp add email varchar2(10);

**To modify the table column:**

* To increase column **lettering**
* alter table emp modify email varchar2(15);

**Rename a table column name:**

* alter table emp rename column email to gmail; []

**To drop a column in table:**

* alter table emp drop column gmail;

**To Rename table:**

* rename emp to demp;

**To check number of records count in table:**

* select count (\*) from tablename;

**To check table size:**

* desc dba\_segments
* select sum(bytes/1024/1024) from dba\_segments where owner=‘username’ and segment\_name=‘tablename’;

**Truncate table:**

* if we truncate a table it will delete all the records in the table .and table will present.and the size of deleted records will given back to OS.
* truncate table demp ;

**Delete table:**

* if we delete a table. it delete records in it and table and won’t give back the storage to OS.it will delete meta data.
* delete demp;
* commit;

(OR)

* delete from emp where eno= 1; to delete employe data

**Drop table AND restore dropped table:**

* drop table t1;
* if we drop a table it will go to recycle bin. And we can restore this table.
* flashback table t1 to before drop;

**If we want to drop a table permanently**:

* drop table t1 purge;

**Invisible column in table from 12c:**

* while creating table we can set ‘invisible ‘ option to any column.
* alter table t1 modify (Sal visible);
* Alter table t1 modify (Sal invisible);

**To check table created script:**

* Select dbms\_metadata.get\_ddl(‘TABLE’,’T1’,’U1’) from dual;

**Online Migration of Table Partition or Sub-Partition**

* In Oracle, table partitioning divides a large table into smaller, more manageable pieces called partitions, enhancing performance, data management, and availability by allowing for targeted operations and efficient storage.

**Purpose:**

* Partitioning improves database performance, simplifies data management, and enhances data availability by allowing for targeted operations on specific subsets of data.

**Partitioning Methods:**

* **Range Partitioning:** Maps data to partitions based on ranges of values of the partitioning key.
* **Hash Partitioning:** Maps data to partitions based on a hashing algorithm, ensuring even distribution of rows.
* **List Partitioning:** Enables explicit control over how rows map to partitions by specifying a list of discrete values for the partitioning key.