DATA DEFINITION LANGUAGE

1. Introduction to Data Definition Language (DDL)

DDL statements are used to define or modify the database objects such as tables, views etc.. All DDL statements are auto committed which means the changes will become permanent once executed.

Commonly used DDL statements are:

- CREATE
- ALTER
- DROP
- RENAME
- TRUNCATE

2. Data types

A data type identifies or classifies a particular type of information or data.

Some commonly used data types are:

- **CHAR (size)** - Used to store character strings values of fixed length.
- **VARCHAR2 (size)** – Used to store variable length string data.
- **NUMBER (size, precision)** – Used to store numbers(fixed or floating point)
- **DATE** – Used to represent date and time.
- **LONG** – Used to store large variable length strings (upto 2GB).

3. Creating Table using CREATE

The CREATE keyword is used for creating database objects like tables, views, triggers, and indexes.

Syntax:

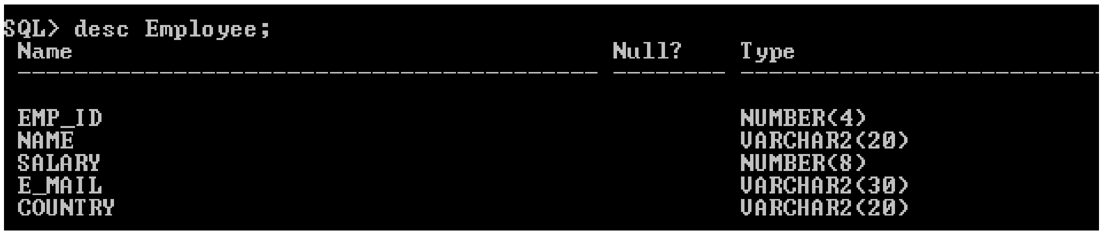
```
CREATE TABLE table_name
(
column_name1 DATATYPE(Size) ,
column_name2 DATATYPE(Size),
column_name3 DATATYPE(Size)
);
```

```
Ex: Create Table Employee
 (
 Emp_id number(4),
 Name varchar2(20),
 Salary number(8),
 E_Mail varchar2(30),
Country varchar2(20)
 );
```

Table created.

The above statement creates a table named Employee with columns Emp_id, Name,Salary,E_Mail and Country.

DESC command can be used to describe the column definitions for the specified table.

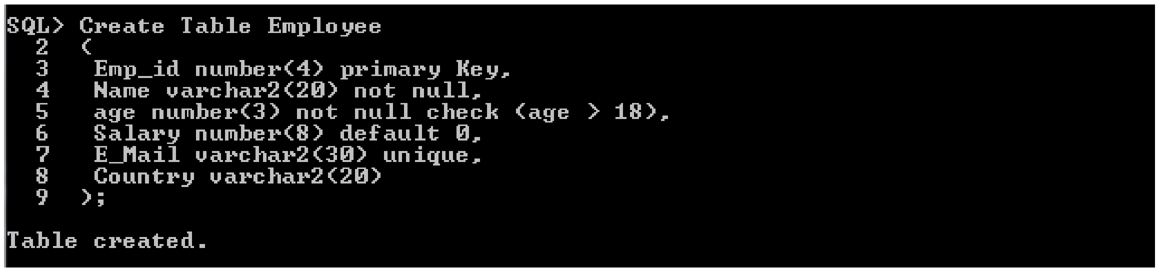


```
SQL> desc Employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 EMP_ID                                             NUMBER(4)
 NAME                                               VARCHAR2(20)
 SALARY                                             NUMBER(8)
 E_MAIL                                             VARCHAR2(30)
 COUNTRY                                            VARCHAR2(20)
```

4. Creating Table with Constraints

The constraints can be created along with creating the table.

Figure 1 shows an example for creating a table with constraints specified on different columns



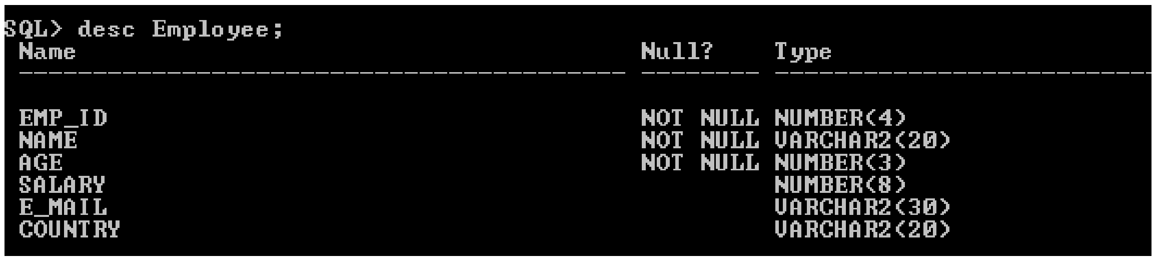
```
SQL> Create Table Employee
  2  (
  3   Emp_id number(4) primary Key,
  4   Name varchar2(20) not null,
  5   age number(3) not null check (age > 18),
  6   Salary number(8) default 0,
  7   E_Mail varchar2(30) unique,
  8   Country varchar2(20)
  9  );

Table created.
```

Figure 1 : Employee table created with constraints

The following are the restrictions created on different column values of Employee table with the help of constraints

- Emp_id – allows only unique and not null values
- Name – not null values
- Age – Not null and value greater than 18
- Salary – If salary value is not specified, then the default value will be 0
- E_Mail – unique values but can have null values

The description of Employee table shown in Figure 2 specifies the columns which cannot have null values.



```
SQL> desc Employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 EMP_ID                                    NOT NULL NUMBER(4)
 NAME                                      NOT NULL VARCHAR2(20)
 AGE                                       NOT NULL NUMBER(3)
 SALARY                                             NUMBER(8)
 E_MAIL                                             VARCHAR2(30)
 COUNTRY                                            VARCHAR2(20)
```

Figure 2: Employee Table Description

Constraints can also be created after creating the table with the help of ALTER command which would be discussed in the upcoming sections.

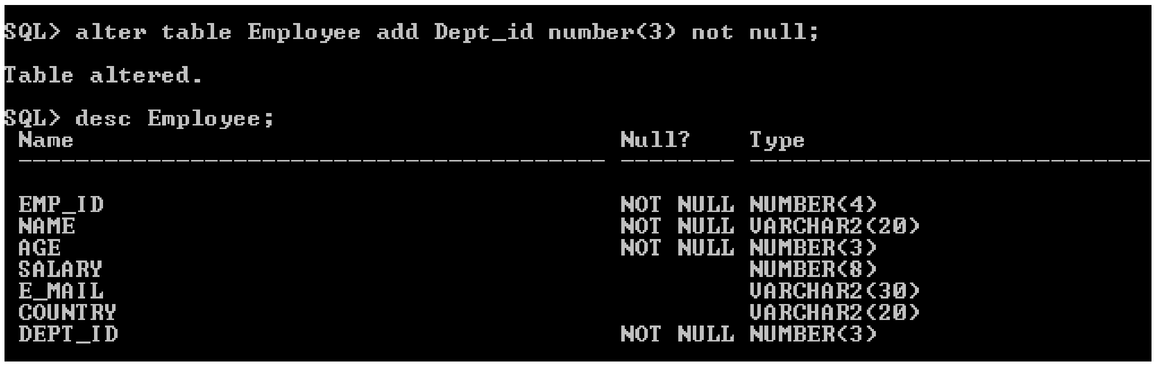
5. ALTER command

ALTER statement is used to modify the structure of database or database objects like tables, views etc..

- **Add a new column to the table**

Syntax:

ALTER TABLE table_name ADD column_name datatype ;
Example:



```
SQL> alter table Employee add Dept_id number(3) not null;

Table altered.

SQL> desc Employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 EMP_ID                                    NOT NULL NUMBER(4)
 NAME                                      NOT NULL VARCHAR2(20)
 AGE                                       NOT NULL NUMBER(3)
 SALARY                                             NUMBER(8)
 E_MAIL                                             VARCHAR2(30)
 COUNTRY                                            VARCHAR2(20)
 DEPT_ID                                   NOT NULL NUMBER(3)
```

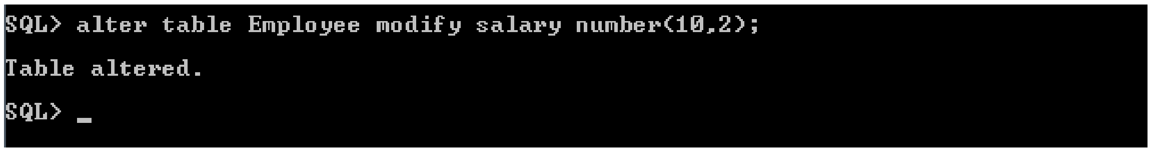
Figure 3 : Adding a new column to Employee table

The ALTER statement in Figure 3 adds a new column 'dept_id' of number data type with constraint not null.

- **Modify an existing column**

Syntax:

ALTER TABLE table_name MODIFY column_name datatype ;



```
SQL> alter table Employee modify salary number(10,2);

Table altered.

SQL> _
```

The above statement modifies the size of salary column from number(8) to number(10,2)

The difference in MYSQL and Oracle standards for modifying a column is shown below:

MYSQL : ALTER TABLE table_name ALTER column_name datatype ;
Oracle : ALTER TABLE table_name MODIFY column_name datatype ;

- **Rename an existing column**

Syntax:

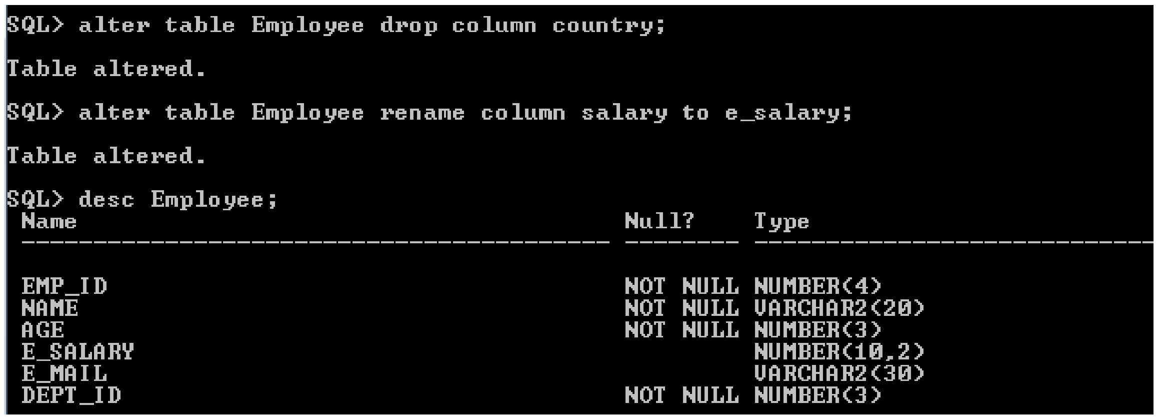
ALTER TABLE table_name RENAME COLUMN old_column_name TO new_column_name ;

In MYSQL, CHANGE keyword is used to rename a column.

- **Delete an existing column**

Syntax:

ALTER TABLE table_name DROP COLUMN column_name ;



```
SQL> alter table Employee drop column country;

Table altered.

SQL> alter table Employee rename column salary to e_salary;

Table altered.

SQL> desc Employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 EMP_ID                                    NOT NULL NUMBER(4)
 NAME                                      NOT NULL VARCHAR2(20)
 AGE                                       NOT NULL NUMBER(3)
 E_SALARY                                           NUMBER(10,2)
 E_MAIL                                             VARCHAR2(30)
 DEPT_ID                                   NOT NULL NUMBER(3)
```

Figure 4: Example for drop and rename column

Figure 4 shows the ALTER statements for deleting the existing column country and to change the column name of salary to e_salary.

- **Adding/Deleting constraints using ALTER**

ALTER command can be used to add or delete a constraint.

Syntax:

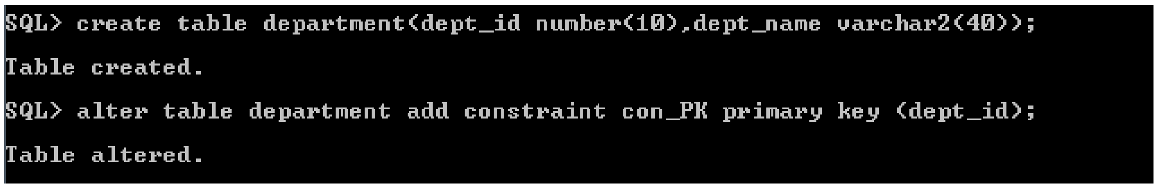
ALTER TABLE table_name ADD CONSTRAINT const_name const_type ;

ALTER TABLE table_name DROP CONSTRAINT const_name ;

In the above statements, const_name and const_type refers to constraint name and constraint type respectively.

Example:

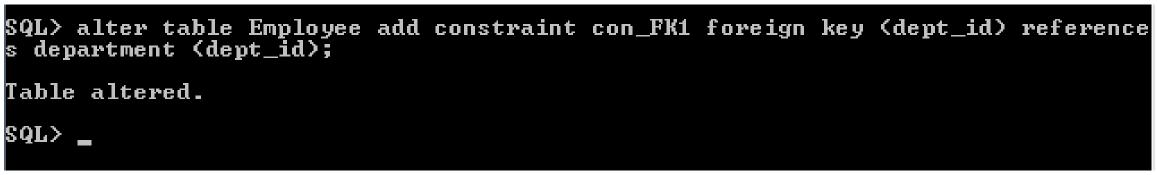
In the example shown in Figure 5, primary key constraint is created on dept_id column for the table Department.



```
SQL> create table department(dept_id number(10),dept_name varchar2(40));

Table created.

SQL> alter table department add constraint con_PK primary key (dept_id);

Table altered.
```

Figure 5: Primary Key creation using ALTER

Foreign Key constraint can be created while creating the table or by ALTER command.



```
SQL> alter table Employee add constraint con_FK1 foreign key (dept_id) reference
s department (dept_id);

Table altered.

SQL> _
```

Figure 6: Foreign Key Creation

In the example shown in Figure:6, Foreign Key constraint named con_FK1 is created on the field dept_id of employee table which refers to the primary key column of department table (dept_id).

Points to remember regarding foreign keys:

- The table containing the primary key can be referenced as the parent table and the table with the corresponding foreign key can be referenced as the child table. The primary key column of parent table and foreign key column of child table should be of the same data type.
- Foreign key column in the child table cannot have values which are not present in the corresponding primary key column . When an SQL operation (insert, delete or update) attempts to change data in such a way that the above rule is compromised, then a referential constraint violation happens. Therefore this need to be taken care while modifying or inserting data to parent or child table.

6. TRUNCATE, DROP, RENAME

Truncate Table :
Removes all the rows from a table and deallocates the space used by the removed rows. This does not remove the table structure from the database.

Syntax:

TRUNCATE TABLE table_name;

Drop table :
Removes the table including the records and the structure entirely from the database.

Syntax:

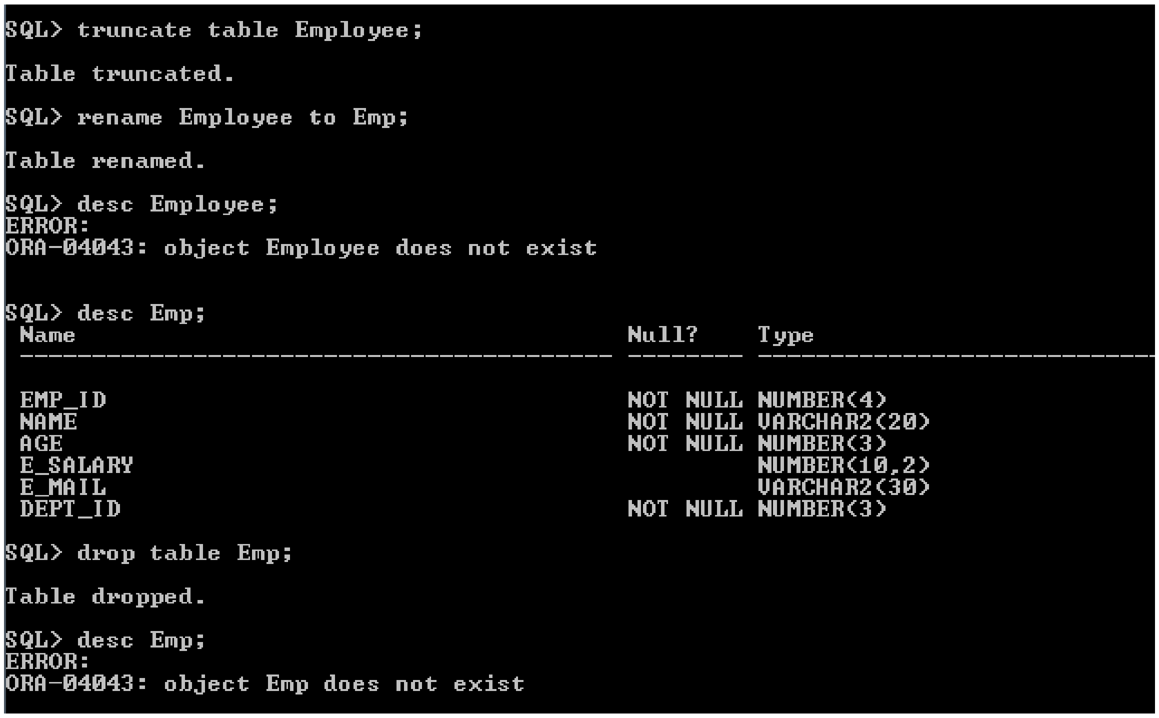
DROP TABLE table_name;

Renaming a Table:
To change the name of an existing table

Syntax:

RENAME old_table_name to new_table_name;

In Figure 5, the Employee table is first truncated and then renamed to Emp. Then Emp table is deleted using drop command.



```
SQL> truncate table Employee;

Table truncated.

SQL> rename Employee to Emp;

Table renamed.

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


SQL> desc Emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 EMP_ID                                    NOT NULL NUMBER(4)
 NAME                                      NOT NULL VARCHAR2(20)
 AGE                                       NOT NULL NUMBER(3)
 E_SALARY                                           NUMBER(10,2)
 E_MAIL                                             VARCHAR2(30)
 DEPT_ID                                   NOT NULL NUMBER(3)

SQL> drop table Emp;

Table dropped.

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

Figure 5 : Examples of truncate, drop and rename