DBMS
Lab Assignment number 3

Name: Aamir Ansari **Batch** A **Roll no.** 01

Aim: Experiment to study Data Definition Language Commands and Integrity constraints.

Theory:

The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project.
Let's take a look at the structure and usage of four basic DDL commands:

CREATE:
Create command is used to create different database objects like table,view etc.
Syntax: create table <table name>(<column name> <datatype>[size],….);

ALTER:
Once you've created a table within a database, you may wish to modify the definition of it. The ALTER command allows you to make changes to the structure of a table without deleting and recreating it. Take a look at the following command:

DROP
The final command of the Data Definition Language, DROP, allows us to remove entire database objects from our DBMS. For example, if we want to permanently remove the personal_info table that we created, we'd use the following command:

Syntax: Drop table tablename;
Integrity constraints.

Integrity constraints:

1. NOT NULL Constraint: When column is defined as a NOT NULL then that column becomes mendatory column i.e value must be entered in that column for each row.
Syntax: columnname datatype(size) NOT NULL

2. Primary key constraint: This constraint ensures that
a)Data entered in the table column is unique across the entire table.
b)None of the cells belonging to the column are left emty.
Syntax: columnname datatype(size) Primary key

3. Unique key constraint: This constraint ensures that
a) Data entered in the table column is unique across the entire table.
b) Column can have NULL value.
Syntax: columnname datatype(size) Unique

4. Foreign key constraint: Foreign key constraint creates relationship between Records. Often we wish to ensure that a value appearing in a relation for a given set of attributes also appears for another set of attributes in another relation. This is called referential integrity This constraint ensures that
a) Record cannot be inserted into a detail if corresponding records in

the master table do not exist.

b) Records of the master table cannot be deleted if corresponding
records in detail table exist.
Syntax: columnname datatype(size) references tablename(columnname)
On delete cascade
5. Check Constraint: Business rule validations can be applied column by using
check constraint. Check constraint must be specified as a logical
expression that evaluates either true or false.
Syntax: columnname datatype(size) CHECK(logical expression)

Defining Integrity Constraints with the CREATE TABLE Command

<u>Example</u>:

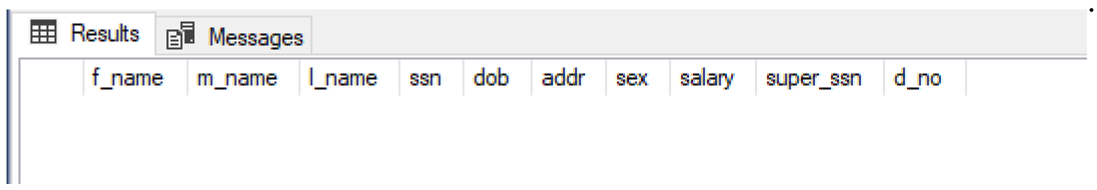
```
--Execution of different types of DDL Commands(create, Alter, Delete)
--create command
        create table customer(cid int, cname varchar(20), address char(10));
--Alter command(add | alter column | drop column)
--adding new column in existing table
        alter table customer add phno numeric(10)
--Modifying existing column
        alter table customer alter column address varchar(20)
--dropping column from existing table
        alter table customer drop column address
--Execution of different types of Integrity constraints
--Table Employee
        create table employee(ssn int primary key check(ssn like &#39;__&#39;),ename varchar(20)
        not null, salary money, superssn int foreign key references employee(ssn),
        dno int default 5)
--Table Department
        create table dept(dno int primary key, dname varchar(20), mgrssn int references
        employee, startdate datetime)
-- applying foreign key constarint on existing table
        alter table employee add constraint fk_dno foreign key(dno) references dept(dno)
--Table deptlocation
        create table deptloc(dno int,dloc varchar(20),primary key(dno,dloc),foreign key(dno)
        references dept(dno))
--Table Project
        create table project(pno int primary key,pname varchar(20),dno int references, dept(dno))
--Table workson
        create table workson(ssn int, pno int, noofhrs int, primary key(ssn,pno),
        foreign key(ssn) references employee(ssn) on delete cascade on update cascade,
        foreign key(pno) references project(pno) on delete cascade on update cascade)
--Table Dependent
        create table dependent(ssn int,depname varchar(20),relation varchar(20),
        primary key(ssn,depname),foreign key(ssn) references employee(ssn))
--droping primary key constraint from existing table
        alter table dependent drop constraint PK__dependent__3B0BC30C
--Applying constraint on existing table
        alter table dependent add constraint pk_ssn primary key(ssn,depname)
```

Code:

```
CREATE TABLE Employee (
	f_name VARCHAR(30) NOT NULL,
	m_name VARCHAR(30),
	l_name VARCHAR(30),
	ssn BIGINT NOT NULL,
	dob DATE,
	addr VARCHAR(50),
	sex CHAR(1),
	salary MONEY,
	super_ssn BIGINT NOT NULL,
	d_no INT,

	PRIMARY KEY (ssn),
	FOREIGN KEY (super_ssn) REFERENCES Employee(ssn)
);

SELECT * FROM Employee;
```



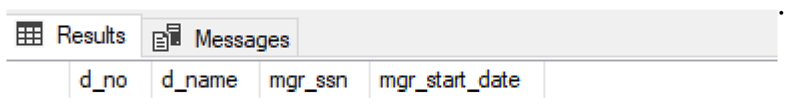
```
f_name  m_name  l_name  ssn  dob  addr  sex  salary  super_ssn  d_no
```

***

```
CREATE TABLE Department (
	d_no INT NOT NULL,
	d_name VARCHAR(20) NOT NULL,
	mgr_ssn BIGINT,
	mgr_start_date DATE,

	PRIMARY KEY (d_no),
	FOREIGN KEY (mgr_ssn) REFERENCES Employee(ssn)
);

SELECT * FROM Department;
```



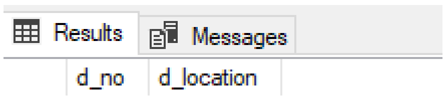
```
d_no  d_name  mgr_ssn  mgr_start_date
```

***

```
ALTER TABLE Employee
ADD FOREIGN KEY (d_no) REFERENCES Department(d_no);
```

```
CREATE TABLE Department_location (
	d_no INT NOT NULL,
	d_location VARCHAR(20) NOT NULL,

	PRIMARY KEY (d_no, d_location),
	FOREIGN KEY (d_no) REFERENCES Department (d_no)
);
SELECT * FROM Department_location;
```



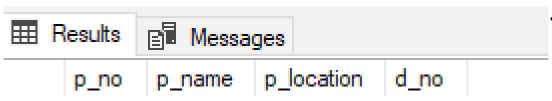
```
   d_no   d_location
```

***

```
CREATE TABLE Project (
	p_no INT NOT NULL,
	p_name VARCHAR(20) NOT NULL,
	p_location VARCHAR(20) NOT NULL,
	d_no INT NOT NULL,

	PRIMARY KEY (p_no),
	FOREIGN KEY (d_no) REFERENCES Department (d_no)
);

SELECT * FROM Project;
```



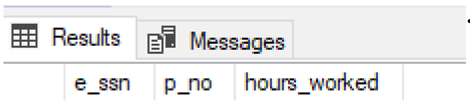
```
   p_no   p_name   p_location   d_no
```

***

```
CREATE TABLE Works_on (
	e_ssn BIGINT NOT NULL,
	p_no INT NOT NULL,
	hours_worked FLOAT,

	PRIMARY KEY (e_ssn, p_no),
	FOREIGN KEY (e_ssn) REFERENCES Employee (ssn),
	FOREIGN KEY (p_no) REFERENCES Project (p_no)
);

SELECT * FROM Works_on;
```

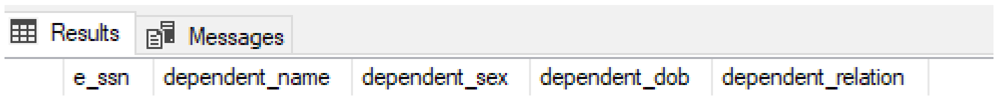


```
   e_ssn   p_no   hours_worked
```

***

```
CREATE TABLE Dependant (
        e_ssn BIGINT NOT NULL,
        dependent_name VARCHAR(30) NOT NULL,
        dependent_sex CHAR(1),
        dependent_dob DATE,
        dependent_relation VARCHAR(20),

        PRIMARY KEY (e_ssn, dependent_name),
        FOREIGN KEY (e_ssn) REFERENCES Employee (ssn)
);
SELECT * FROM Dependant;
```



e_ssn	dependent_name	dependent_sex	dependent_dob	dependent_relation

***

Conclusion: Thus we have implemented different DDL Commands and Integrity constraints successfully.