**K. J. Somaiya College of Engineering, Mumbai-77**
**(Autonomous College Affiliated to University of Mumbai)**

| | |
|---|---|
| **Batch: A2** | **Roll No.: 1911027** |
| **Experiment / assignment / tutorial No. 3** | |
| **Grade: AA / AB / BB / BC / CC / CD /DD** | |

**Title:** Implementation of Database in SQL -DDL

**Objective:** Define/modify database definitions with proper constraints

**Expected Outcome of Experiment:**
CO 2: Convert entity-relationship diagrams into relational tables, populate a relational database and formulate SQL queries on the data Use SQL for creation and query the database.

CO 3: Define and apply integrity constraints and improve database design using normalization techniques.

**Books/ Journals/ Websites referred:**


1. Sharaman Shah,"*Oracle for Professional*", SPD.
2. Dr. P.S. Deshpande, SQL and PL/SQL for Oracle 10g.Black book, Dreamtech Press
3. Korth, Slberchatz, Sudarshan: "Database Systems Concept", 5th Edition , McGraw Hill
4. Peter Rob and Carlos Coronel,"Database Systems Design, Implementation and Management", Thompson Learning, 5th Edition


**Pre Lab/ Prior Concepts**:
Resources used: Postgresql

**Theory:** The set of relations in a database must be specifies to the system by means of a data definition language (DDL). The SQL DDL allows specification of not only a set of relations but also specific information about the relation including,

1. The schema for each relation
2. The domain of values associated with each attribute
3. The integrity constraints
4. The set of indices to be maintained for each relation
5. The security and authorization information for each relation
6. The physical storage structure of each relation on disk

**Syntax Create Table:**

create table employee(ssn,fname varchar(10), mname varchar(10), lname varchar(10),

desg varchar(20), gender varchar(5), addr varchar(20), bdate datetime, sal float,primary

key(ssn));

create table manages(ssn int, dept_code int, start_dt datetime, foreign key(ssn)

create table manages(ssn int, dept_code int, start_dt datetime, foreign key(ssn)

references employee, foreign key(dept_code) refrences department, key(ssn,dept_code) ) on delete set null;primary

**Data Constraints**

Busines managers of the organization determine the a set of rules that must be applied before the data is stored in the database. The application of such rules on raw data ensures **data integrity**.

**Eg:-** An employee belonging to Sales department cannot have salary higher than Rs. 1000.

An employee has an unique identification number.

## Applying Data Constraints

Oracle permits data constraints to be attached to table columns using SQL syntax.

Constraints can be attached to table columns using

Alter table

**Unique Constraint**

**Unique Constraint- At column level Syntax**

**<ColumnName><Datatype>(<size>)**

**UNIQUE Unique Constraint- At table level**

**CREATE TABLE<TableName>(**

**<ColumnName><Datatype>(<size>)**

**<ColumnName><Datatype>(<size>)**

**<Columnname><Datatype>(<size>)**

**UNIQUE(<ColumnName1>,<ColumnName2>);**

**Implementation Details (Problem Statement, Query and Screenshots of Results):**

**Problem Definition:** Now a days as technology is advancing in the healthcare sector so is the intensity and frequency of diseases. So the need for medicine is increasing at a non-linear rate, our pharma sector on the other side was struggling to cope with the accelerating need in the past. The scenario is different now and so is the problem. We currently have a good production rate of these medicines, but as we know as one problem is solved another is ready to make its way.

In this technological era we suppose things to be done as fast as possible. Now consider a person who wants a particular medicine. He goes to some medical shops and inquires about the medicine, there is a good amount of chance that he might find the medicine without wasting much time roaming around 4-5 medical shops. If he finds the medicine in a medical shop say 'x' and the price is say 'a', and he purchases the medicine. And somewhere around him nearby there is another medical store that is providing the same medicine for a discounted rate. The person has not made an optimized purchase. But on the contrary if he does not find the medicine then what? Another case being, if there are only senior citizens staying in a house then, how feasible it is for them to go and purchase medicine each time they need it?

So therefore we came up with an idea to sell medicines and deliver it to a particular customer at his/her door step. The ideas goes like this, we first take information of a particular customer to store it into our database. We then gather a list of all the medicine shops (Mainly major ones to ensure availability of all medicines at any given time). Now our customer could order any medicine he wants through our portal. When he/she does so and searches for a particular medicine our system would first of all list all the medical shops that have this particular medicine available. After which our system would also recommend that particular medical which is providing the medicine at the least price thus giving our customer a better purchase.

There are many advantages of this type of systems like lower prices of medicines, availability of medicines i.e. if medicines are not available in one of the shops than it can be made available to the customer from another shop as well, price comparison of medicine is also possible.

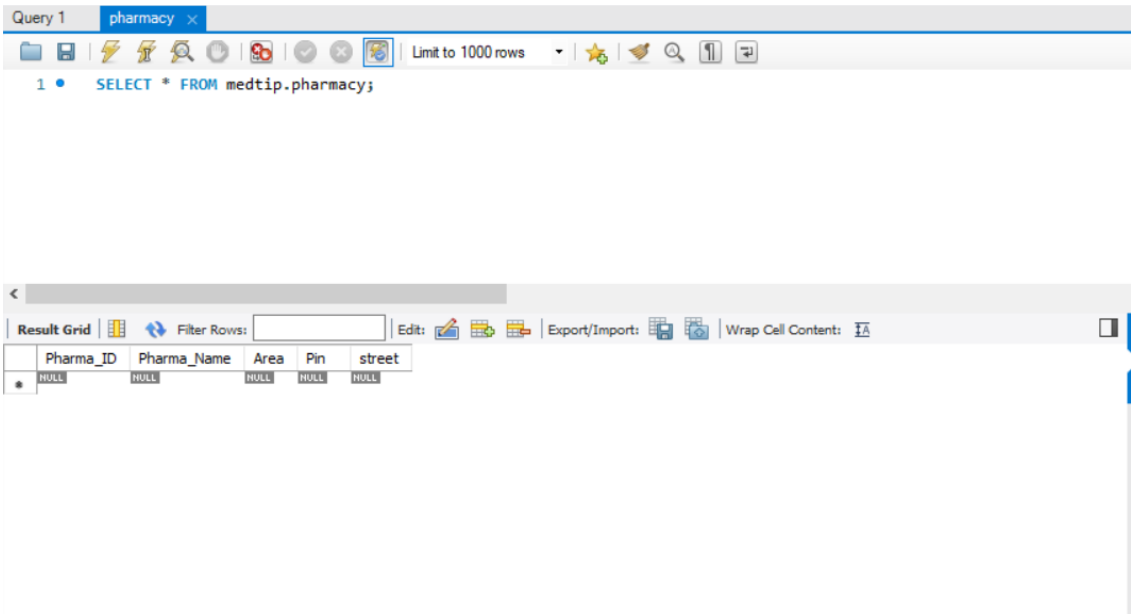
There is no doubt that there are many advantages and benefits of online medicine delivery system but disadvantages are also there. There are many illiterate people who can’t do such things efficiently. There are many fake prescription related issues that are also present in the current scenario which would adversely affect the health of people. By ordering medicines online some additional costs are also added to the final amount.

**Queries and Screenshots : -**

**Query : - Pharmacy table**

```
use medtip;
create table Pharmacy(
Pharma_ID int primary key,
Pharma_Name varchar(30) not null unique,
Area varchar(30),
Pin int not null,
street varchar(30)
);
```

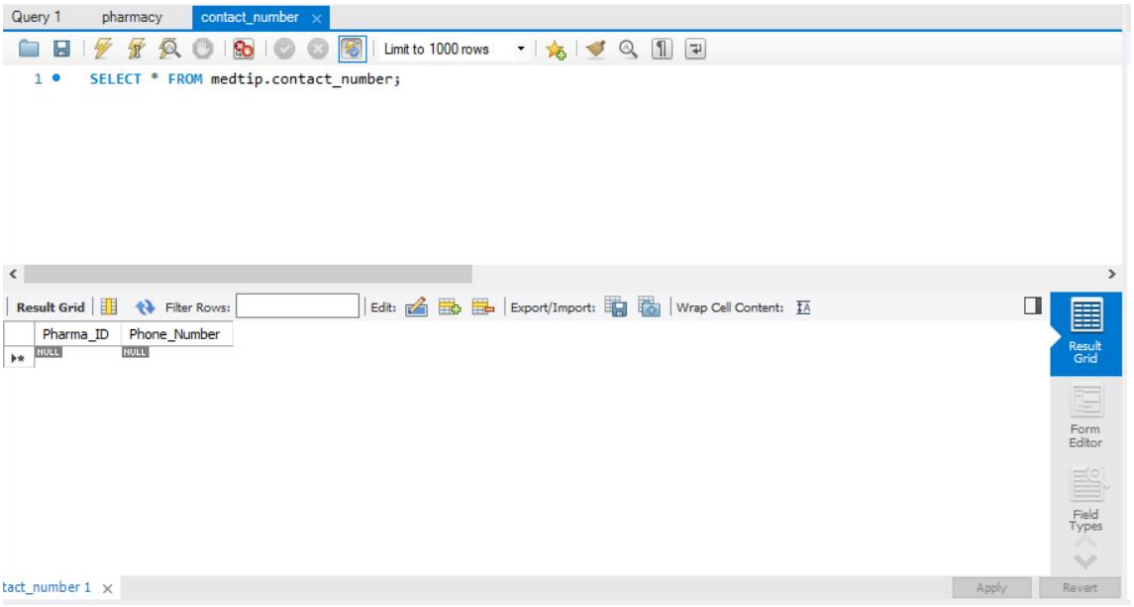
**Screenshot : -**



**Query : - Contact_Number table**

```
create table Contact_Number(
Pharma_ID int,
Phone_Number int,
FOREIGN KEY (Pharma_ID) REFERENCES pharmacy(Pharma_ID),
PRIMARY KEY (Pharma_ID, Phone_Number)
);
```

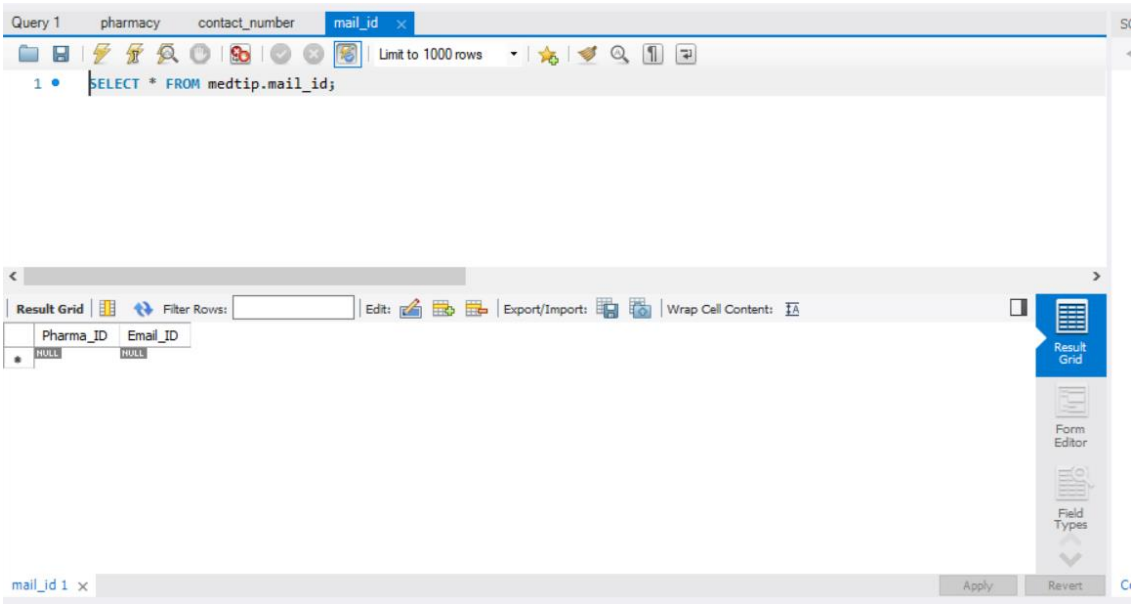
**Screenshot : -**



**Query : - Mail_ID table**

```
create table Mail_ID(
Pharma_ID int,
Email_ID varchar(30),
FOREIGN KEY (Pharma_ID) REFERENCES pharmacy(Pharma_ID),
PRIMARY KEY (Pharma_ID, Email_ID)
);
```

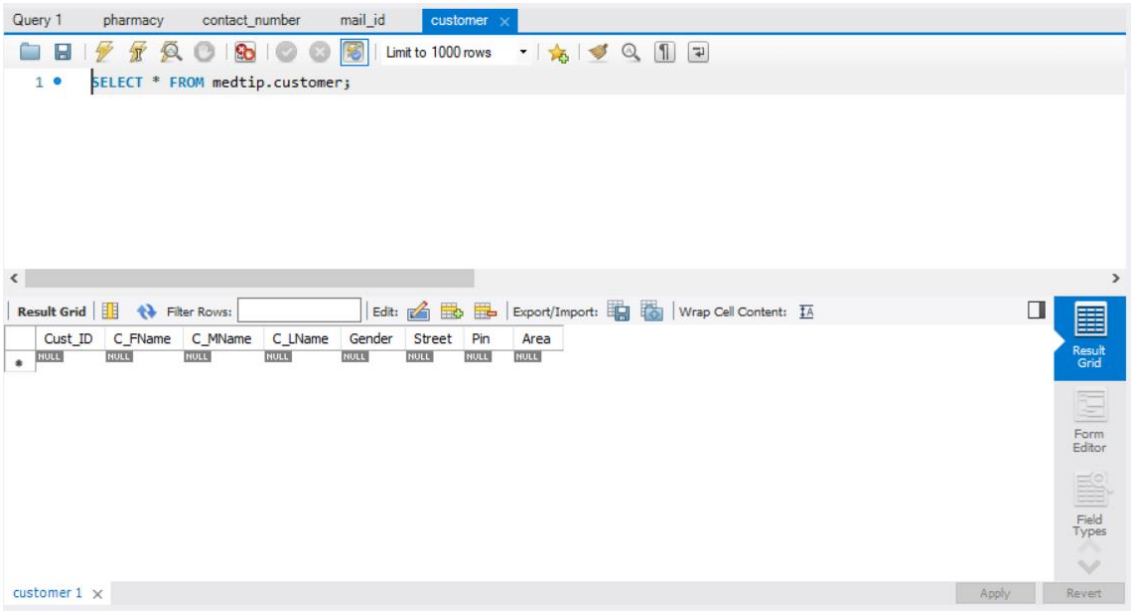
**Screenshot : -**



**Query : - Customer table**

```
create table Customer(
Cust_ID int primary key,
C_FName varchar(12) not null,
C_MName varchar(12),
C_LName varchar(20),
Gender boolean,
Street varchar(30) not null,
Pin int not null,
Area varchar(30)
);
```

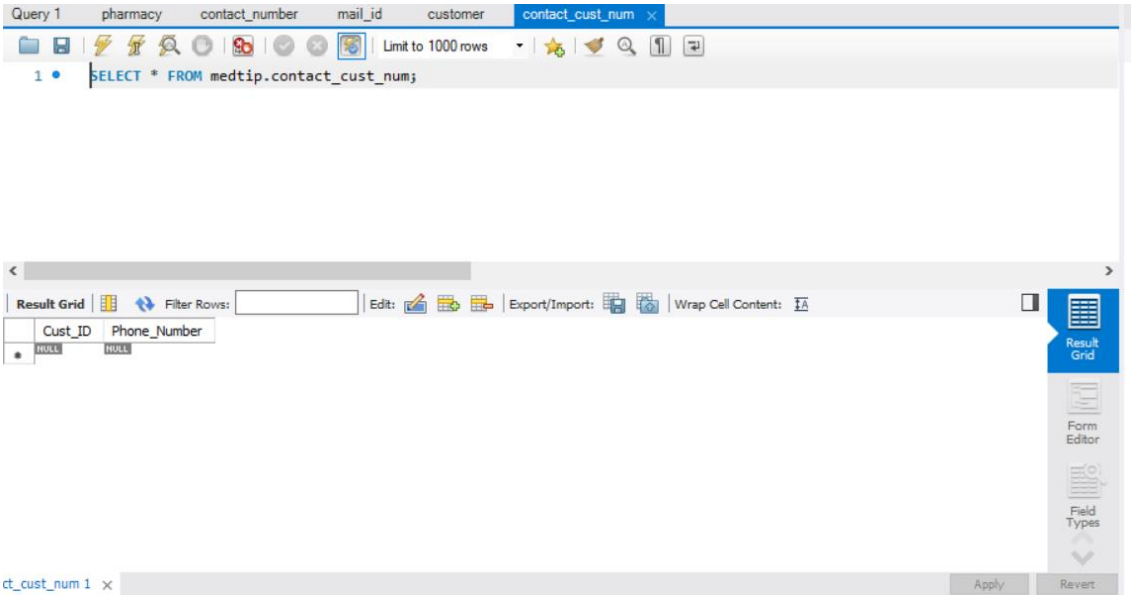
**Screenshot : -**



**Query : - Contact_cust_num table**

```
create table Contact_cust_num(
Cust_ID int,
Phone_Number int,
FOREIGN KEY (Cust_ID) REFERENCES customer(Cust_ID),
PRIMARY KEY (Cust_ID, Phone_Number)
);
```

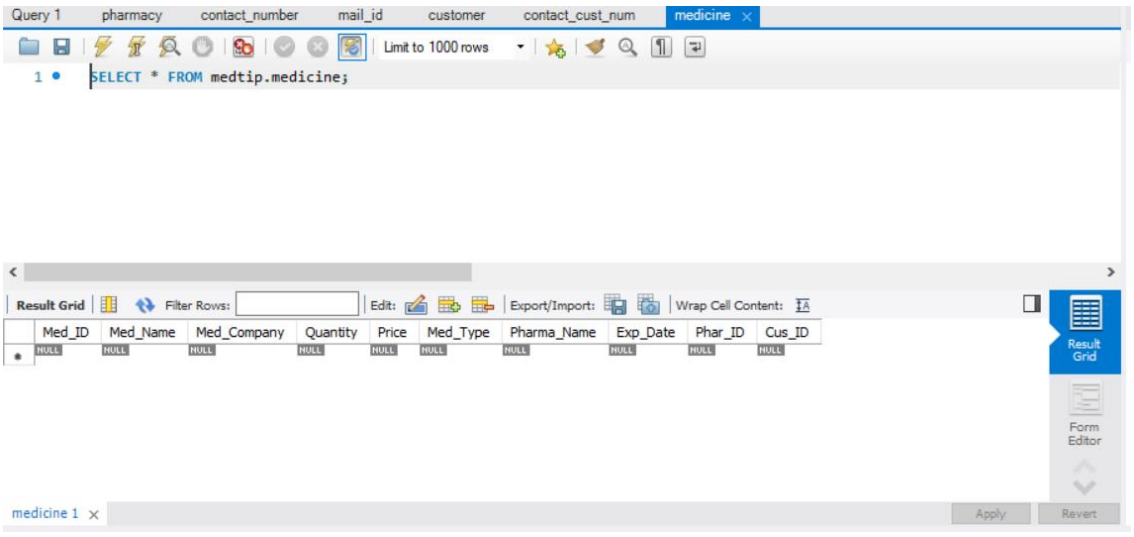
**Screenshot : -**



**Query : - Medicine table**

```
create table Medicine(
Med_ID int primary key,
Med_Name varchar(30) not null unique,
Med_Company varchar(30),
Quantity int default(1),
Price float,
Med_Type varchar(30) not null,
Pharma_Name varchar(30) not null,
Exp_Date date,
Phar_ID int,
Cus_ID int,
FOREIGN KEY (Cus_ID) REFERENCES customer(Cust_ID),
FOREIGN KEY (Phar_ID) REFERENCES pharmacy(Pharma_ID)
);
```

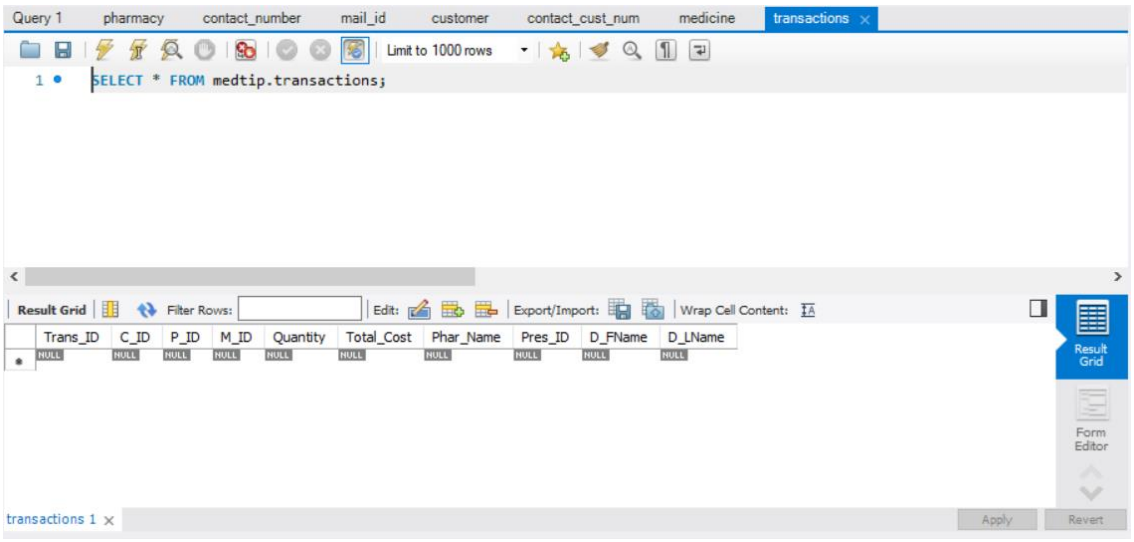
**Screenshot : -**



**Query : - Transactions table**

```
create table Transactions(
Trans_ID int primary key,
C_ID int,
P_ID int,
M_ID int,
Quantity int default(1),
Total_Cost float not null,
Phar_Name varchar(30),
Pres_ID int,
D_FName varchar(30),
D_LName varchar(30),
FOREIGN KEY (C_ID) REFERENCES customer(Cust_ID),
FOREIGN KEY (P_ID) REFERENCES pharmacy(Pharma_ID),
FOREIGN KEY (M_ID) REFERENCES medicine(Med_ID)
);
```

**Screenshot : -**




**Conclusion:** By performing this experiment understood how to make tables in MySql with proper integrity constraints. Also understood how to convert relational model to actual tables in MySql. Successfully created all the tables needed for the project. Also tables are satisfying all integrity constraints.

**Post Lab Questions:**

1. **Which command is used for removing a table and all its data from the database:**

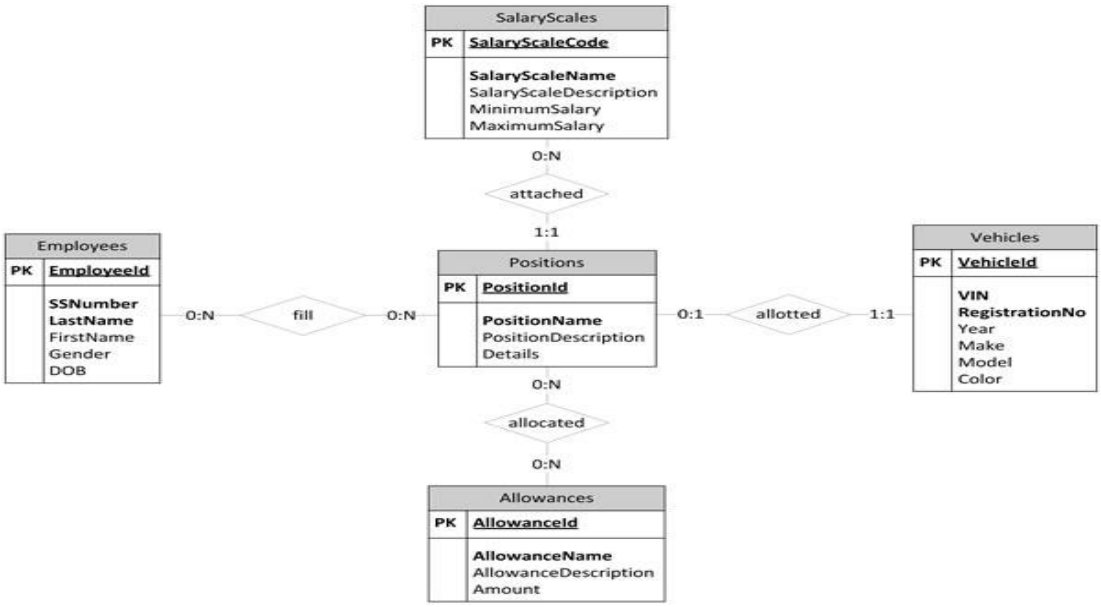
   A. DROP Command
   B. TRUNCATE Command
   C. Both Commands

**ANS) A. DROP Command**

**2. For the given ER model, using DDL command: Write syntax to create CREATE Tables with all possible integrity constraints**

**Problem Statement:**

A small accounting firm wants a simple HR application that will help it to keep track of its employees, their positions, allowances, salary scales, and which company vehicles their employees drive. The application must keep track of all the positions at the firm, the employees filling these positions, the allowances for these positions, the salary scales for these positions, and the company vehicles assigned to these positions.




**ANS)**

**--Employees table**

```
CREATE TABLE Employees (

  EmployeeId int(11) NOT NULL AUTO_INCREMENT,

  SSNumber varchar(11) NOT NULL,
```

```
  LastName varchar(50) NOT NULL,
  FirstName varchar(50) NOT NULL,
  MiddleName varchar(50) DEFAULT NULL,
  Gender varchar(6) NOT NULL DEFAULT 'MALE',
  DOB date NOT NULL,
  Email varchar(75) NOT NULL,
  Mobile varchar(14) DEFAULT NULL,
  HTel varchar(14) NOT NULL,
  AddressLine1 varchar(75) NOT NULL,
  AddressLine2 varchar(75) DEFAULT NULL,
  City varchar(50) NOT NULL,
  State varchar(50) NOT NULL,
  PostCode varchar(10) NOT NULL,
  PRIMARY KEY (EmployeeId),
  INDEX SSNumber (SSNumber),
  INDEX DOB (DOB),
  INDEX PostCode (PostCode),
  INDEX FullName (LastName,FirstName)
);
```

**--Allowances table**

```
CREATE TABLE Allowances (
  AllowanceId int(11) NOT NULL AUTO_INCREMENT,
```

```
  AllowanceName int(11) NOT NULL,
  AllowanceDescription varchar(250) DEFAULT NULL,
  Amount decimal(10,2) NOT NULL DEFAULT '0.00',
  PRIMARY KEY (AllowanceId),
  INDEX AllowanceName (AllowanceName)
);
```

**--SalaryScales table**

```
CREATE TABLE SalaryScales (
  SalaryScaleCode tinyint(4) NOT NULL,
  SalaryScaleName varchar(50) NOT NULL,
  SalaryScaleDescription varchar(250) DEFAULT NULL,
  MinimumSalary decimal(10,2) NOT NULL DEFAULT '0.00',
  MaximumSalary decimal(10,2) NOT NULL DEFAULT '0.00',
  PRIMARY KEY (SalaryScaleCode),
  INDEX SalaryScaleName (SalaryScaleName)
);
```

**--Positions table**

```
CREATE TABLE Positions (
  PositionId int(11) NOT NULL AUTO_INCREMENT,
  PositionName varchar(75) NOT NULL,
  PositionDesctiption varchar(250) DEFAULT NULL,
```

```
  Details text,
  SalaryScaleCode tinyint(4) NOT NULL,
  PRIMARY KEY (PositionId),
  FOREIGN KEY SalaryScaleCode (SalaryScaleCode)
        REFERENCES SalaryScales (SalaryScaleCode)
        ON UPDATE CASCADE ON DELETE RESTRICT,
  INDEX PositionName (PositionName)
);
```

**--Vehicles table**

```
CREATE TABLE Vehicles (
  VehicleId int(11) NOT NULL AUTO_INCREMENT,
  VIN varchar(17) NOT NULL,
  RegistrationNo varchar(10) NOT NULL,
  Year year(4) DEFAULT NULL,
  Make varchar(25) DEFAULT NULL,
  Model varchar(25) DEFAULT NULL,
  Color varchar(25) DEFAULT NULL,
  PositionId int(11) NOT NULL,
  PRIMARY KEY (VehicleId),
  FOREIGN KEY PositionId (PositionId)
        REFERENCES Positions (PositionId)
        ON UPDATE CASCADE ON DELETE RESTRICT,
```

```
  INDEX VIN (VIN),
  INDEX RegistrationNo (RegistrationNo)
);
```

**--PositionAllowances table**

```
CREATE TABLE PositionAllowances (
  PosAllowId int(11) NOT NULL AUTO_INCREMENT,
  AllowanceId int(11) NOT NULL,
  PositionId int(11) NOT NULL,
  PRIMARY KEY (PosAllowId),
  FOREIGN KEY AllowanceId (AllowanceId)
        REFERENCES Allowances (AllowanceId)
        ON UPDATE CASCADE ON DELETE RESTRICT,
  FOREIGN KEY PositionId (PositionId)
        REFERENCES Positions (PositionId)
        ON UPDATE CASCADE ON DELETE RESTRICT
);
```

**--EmployeePositions**

```
CREATE TABLE EmployeePositions (
  EmpPosId int(11) NOT NULL AUTO_INCREMENT,
  EmployeeId int(11) NOT NULL,
  PositionId int(11) NOT NULL,
```

```
  StartDate date NOT NULL,
  EndDate date DEFAULT NULL,
  Comments text,
  PRIMARY KEY (EmpPosId),
  FOREIGN KEY EmployeeId (EmployeeId)
        REFERENCES Employees (EmployeeId)
        ON UPDATE CASCADE ON DELETE RESTRICT,
  FOREIGN KEY PositionId (PositionId)
        REFERENCES Positions (PositionId)
        ON UPDATE CASCADE ON DELETE RESTRICT,
  INDEX StartDate (StartDate)
);
```