**Batch: B-1 Roll No.: 16010122104**

**Experiment / assignment / tutorial No. 3**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **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.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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) references employee, foreign key(dept\_code) references department, key(ssn,dept\_code) ) on delete set null

**Data Constraints**

Business managers of the organization determine 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 the Sales department cannot have a 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**

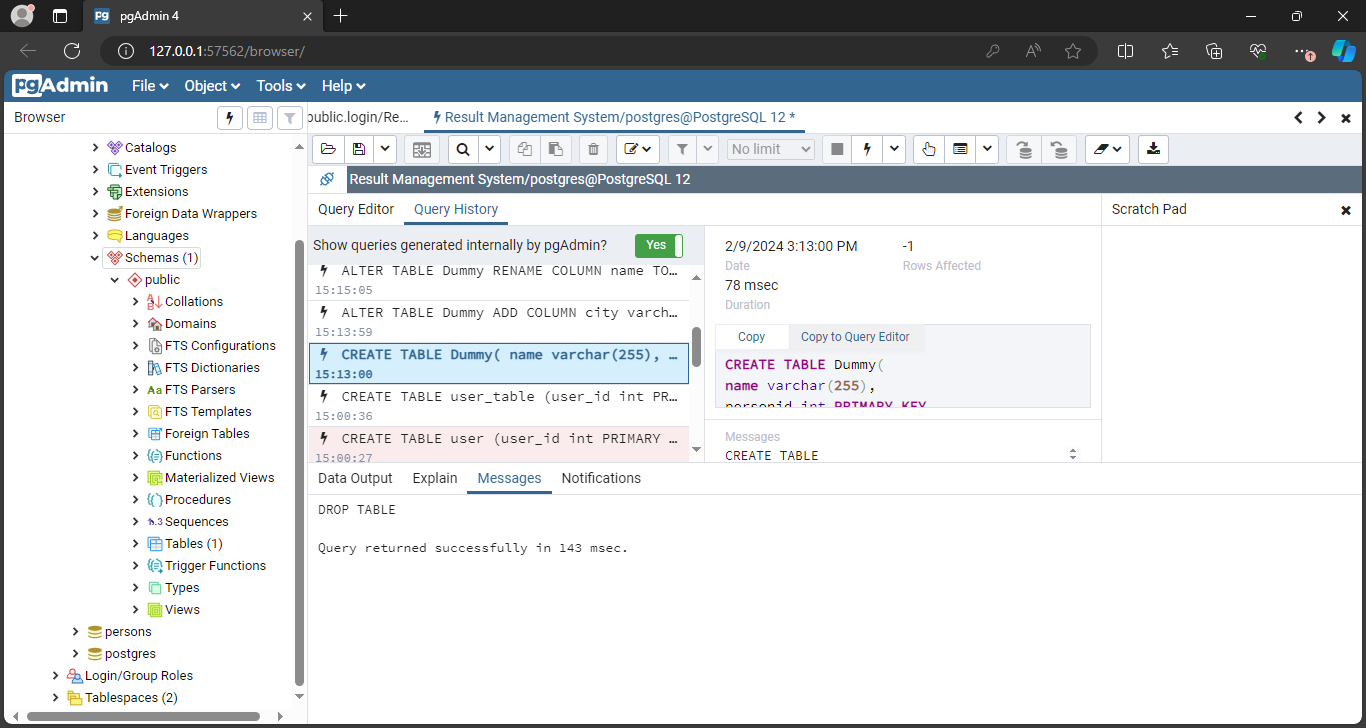
**CREATE TABLE<TableName>(**

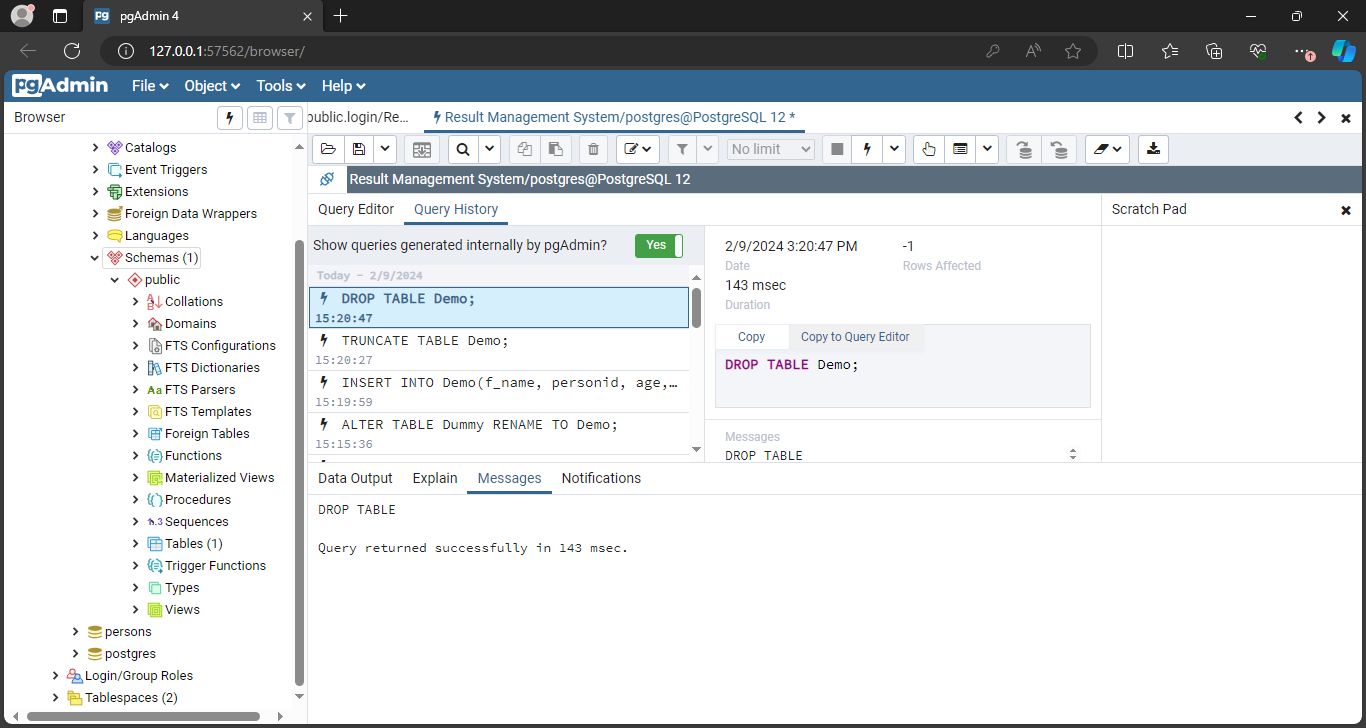
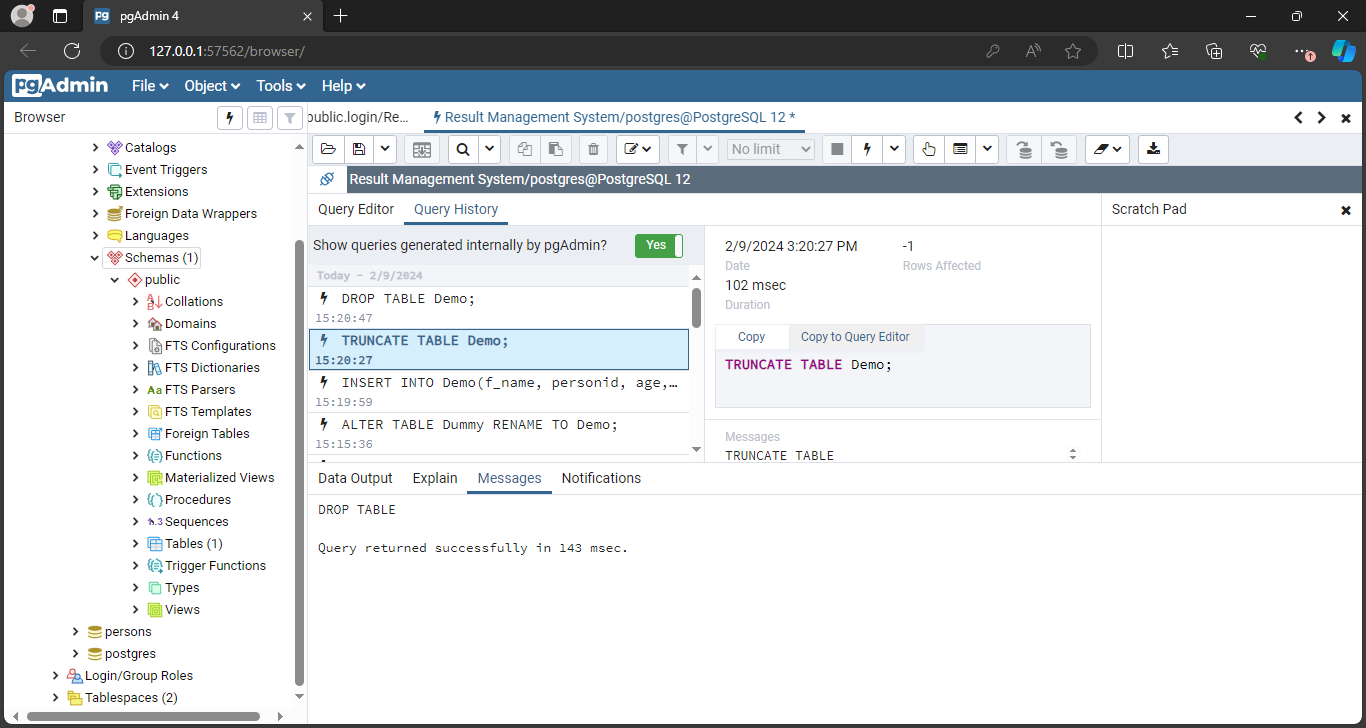
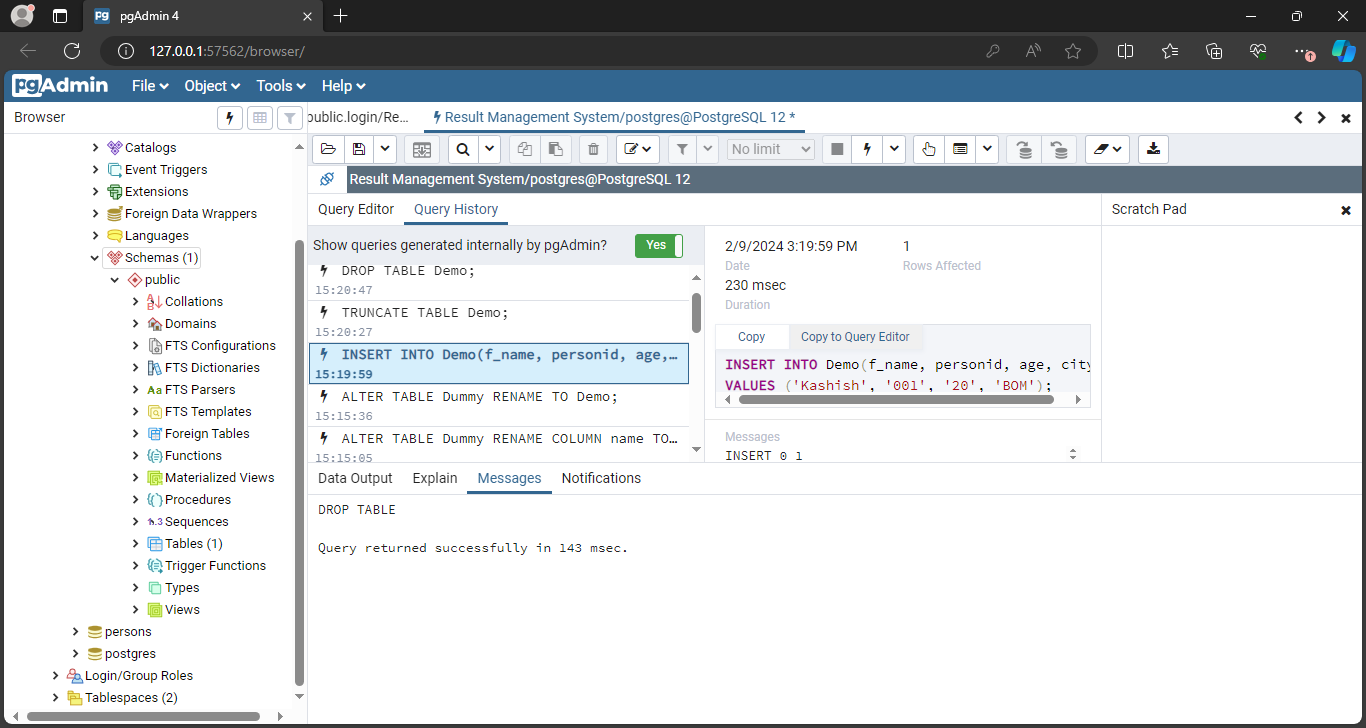
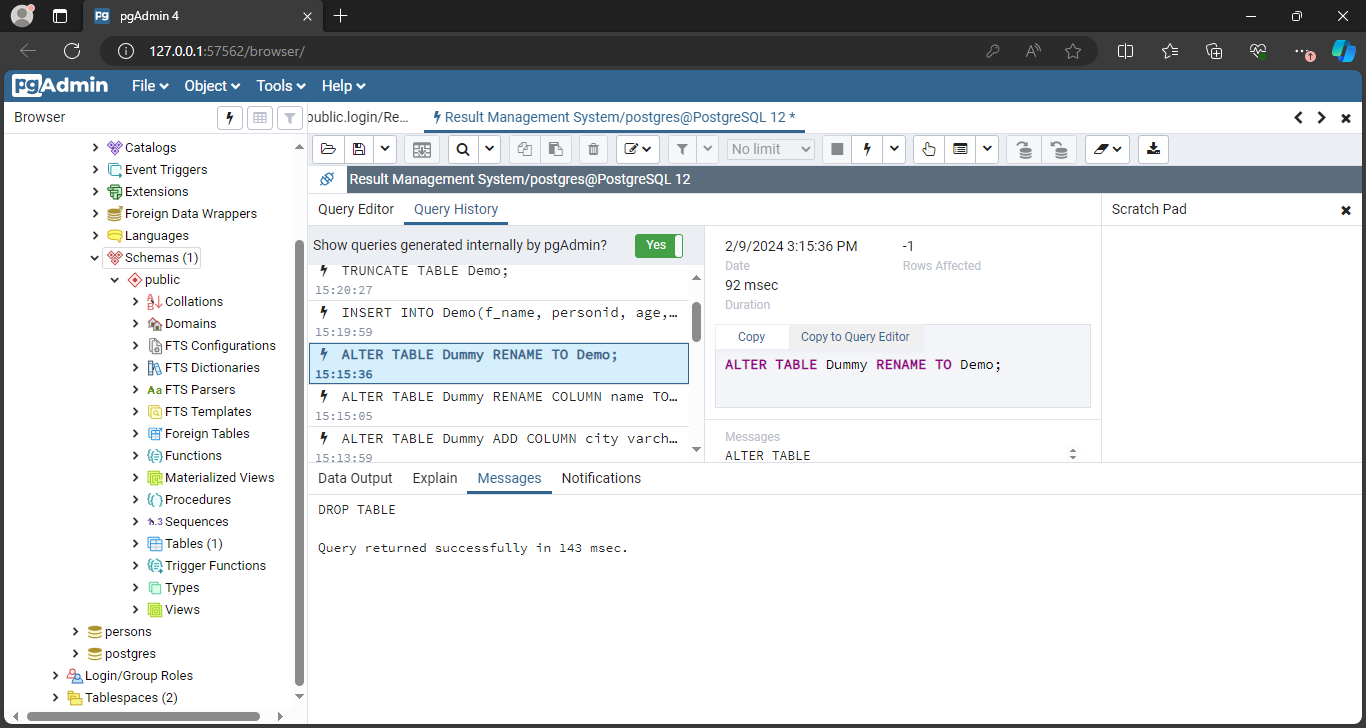
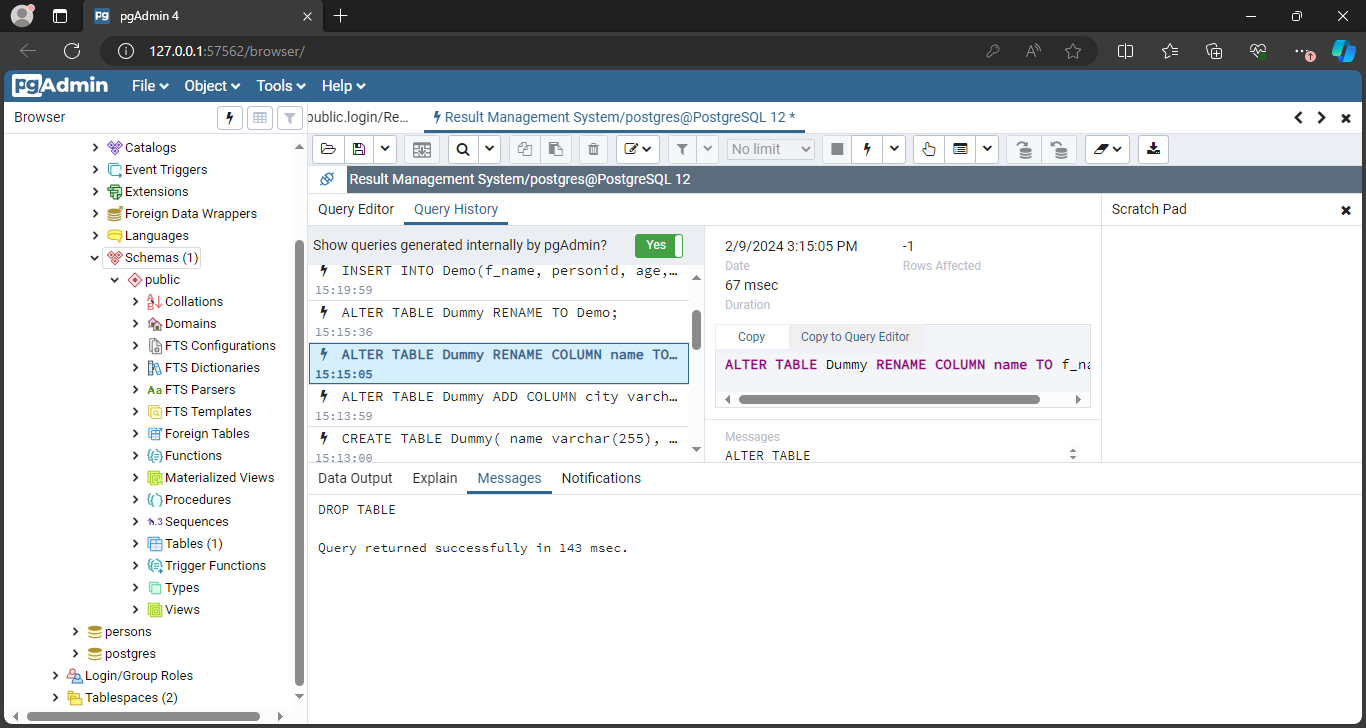
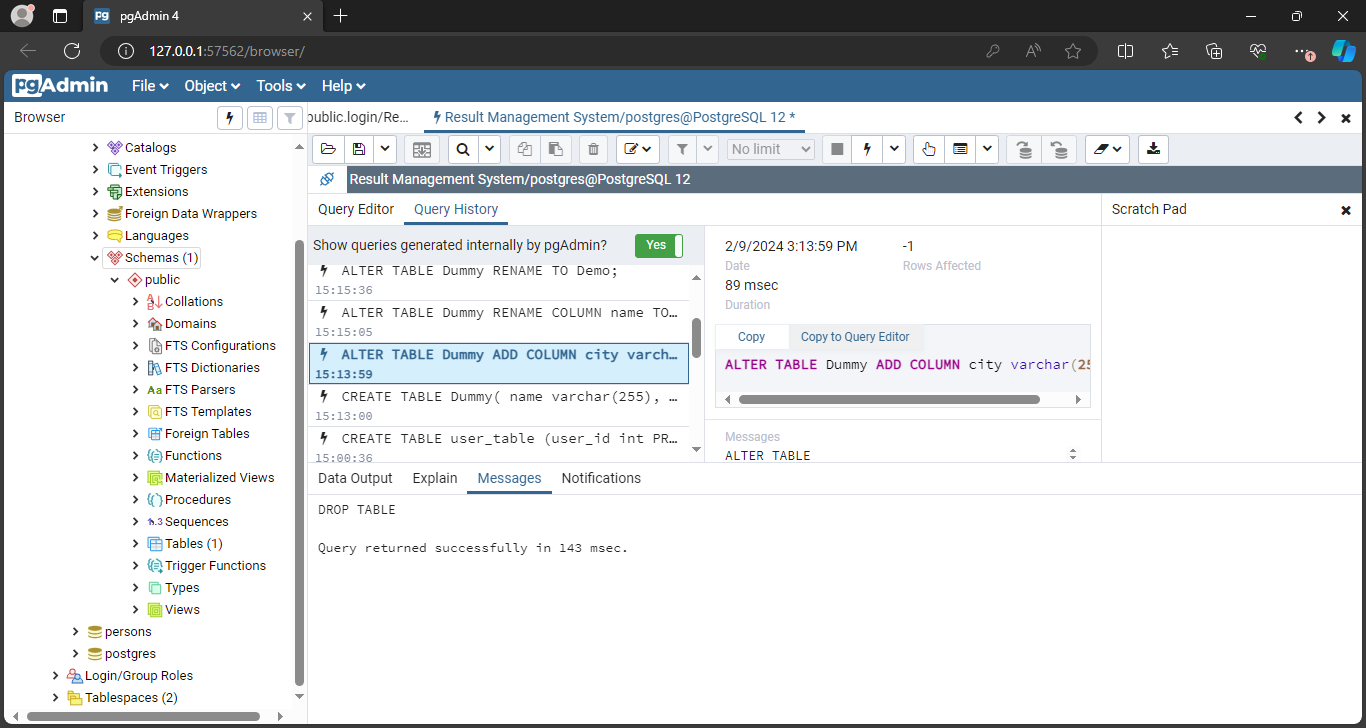
**<ColumnName><Datatype>(<size>)**

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

**<Columnname><Datatype>(<size>) UNIQUE(<ColumnName1>,<ColumnName2>);**

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

****



**Post Lab Questions:**

1. **Explain in brief the following terms:**
2. **Database**
3. **Types of databases**
4. **SQL Data Types**
5. **foreignkey**

**Ans:** Explanation of Terms:

a. Database:  
A database is a structured collection of data that is organized and stored electronically. It is designed to efficiently manage, retrieve, and manipulate data according to the requirements of various applications. Databases typically consist of one or more tables, each containing rows and columns of data, and they are managed by database management systems (DBMS). Databases are used in various fields such as business, education, healthcare, finance, and more to store and manage information.

b. Types of Databases:  
There are several types of databases, each designed to handle specific types of data and applications. Some common types include:

* Relational Databases: Organize data into tables with predefined relationships between them, managed by relational database management systems (RDBMS) like MySQL, PostgreSQL, Oracle, etc.
* NoSQL Databases: Designed to handle unstructured or semi-structured data, providing flexibility and scalability, examples include MongoDB, Cassandra, Redis, etc.
* Object-Oriented Databases: Store data in the form of objects, offering better support for complex data structures and relationships.
* Graph Databases: Optimize for data with complex relationships, using graph structures to represent and query data efficiently.
* Distributed Databases: Spread data across multiple nodes or servers for improved performance, fault tolerance, and scalability.

c. SQL Data Types:  
SQL (Structured Query Language) data types define the type of data that can be stored in a column of a database table. Some common SQL data types include:

* Numeric Types: INT, FLOAT, DECIMAL, etc., for storing numeric values.
* Character String Types: VARCHAR, CHAR, TEXT, etc., for storing text or character data.
* Date and Time Types: DATE, TIME, DATETIME, TIMESTAMP, etc., for storing date and time values.
* Boolean Type: BOOLEAN, for storing true/false values.
* Binary Types: BLOB, BYTEA, etc., for storing binary data.
* Custom Types: ENUM, ARRAY, JSON, etc., for storing custom data types or structured data.

d. Foreign Key:  
A foreign key is a column or a set of columns in a table that establishes a relationship with a primary key or a unique key in another table. It enforces referential integrity by ensuring that values in the foreign key column(s) correspond to values in the referenced primary key or unique key column(s) in the related table. Foreign keys are used to establish and maintain relationships between tables in a relational database, enabling data consistency and integrity.

1. **What are the commands to:**
2. **Delete an entire table.**
3. **To view a database.**
4. **To select & view all the columns.**

**Ans: Commands:**

* 1. **Delete an entire table:**

DROP TABLE table\_name;

* 1. **To view a database:**

SHOW DATABASES;

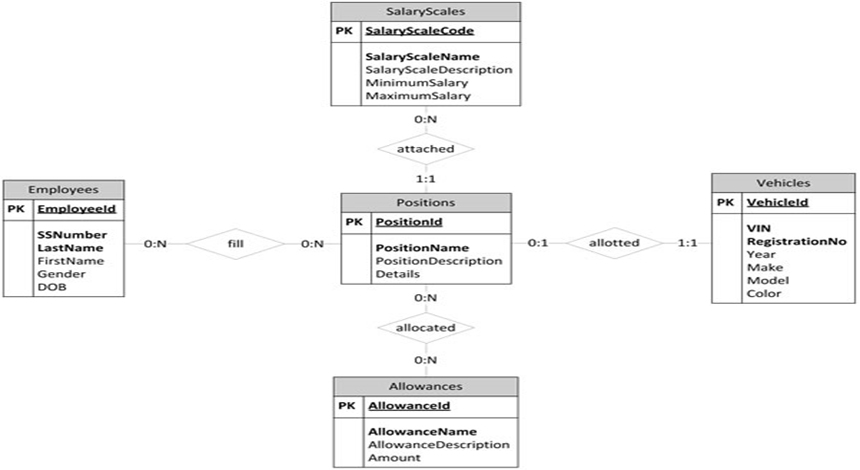
* 1. **To select & view all the columns:**

SELECT \* FROM table\_name;

1. **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.



To create tables for the given ER model with all possible integrity constraints, we'll need to translate the entities and relationships into SQL DDL (Data Definition Language) commands. Below is the syntax to create the tables with integrity constraints:

-- Table for Positions

CREATE TABLE Positions (

position\_id INT PRIMARY KEY,

position\_name VARCHAR(50) NOT NULL,

UNIQUE (position\_name)

);

-- Table for Employees

CREATE TABLE Employees (

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR(100) NOT NULL,

position\_id INT,

FOREIGN KEY (position\_id) REFERENCES Positions(position\_id)

);

-- Table for Allowances

CREATE TABLE Allowances (

allowance\_id INT PRIMARY KEY,

allowance\_name VARCHAR(50) NOT NULL,

amount DECIMAL(10, 2) NOT NULL

);

-- Table for Salary Scales

CREATE TABLE SalaryScales (

scale\_id INT PRIMARY KEY,

position\_id INT,

scale\_name VARCHAR(50) NOT NULL,

min\_salary DECIMAL(10, 2) NOT NULL,

max\_salary DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (position\_id) REFERENCES Positions(position\_id)

);

-- Table for Company Vehicles

CREATE TABLE CompanyVehicles (

vehicle\_id INT PRIMARY KEY,

vehicle\_name VARCHAR(50) NOT NULL,

registration\_number VARCHAR(20) NOT NULL UNIQUE

);

-- Table to represent relationship between Employees and Company Vehicles

CREATE TABLE EmployeeVehicles (

employee\_id INT,

vehicle\_id INT,

PRIMARY KEY (employee\_id, vehicle\_id),

FOREIGN KEY (employee\_id) REFERENCES Employees(employee\_id),

FOREIGN KEY (vehicle\_id) REFERENCES CompanyVehicles(vehicle\_id)

);

**Conclusion:**

This experiment enabled a systematic understanding of Data requirements of application and EER diagram design, laying the groundwork for effective and optimized database development