



Batch: E-2

Roll No.: 16010123325

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

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025



Theory: The set of relations in a database must be specified 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>)

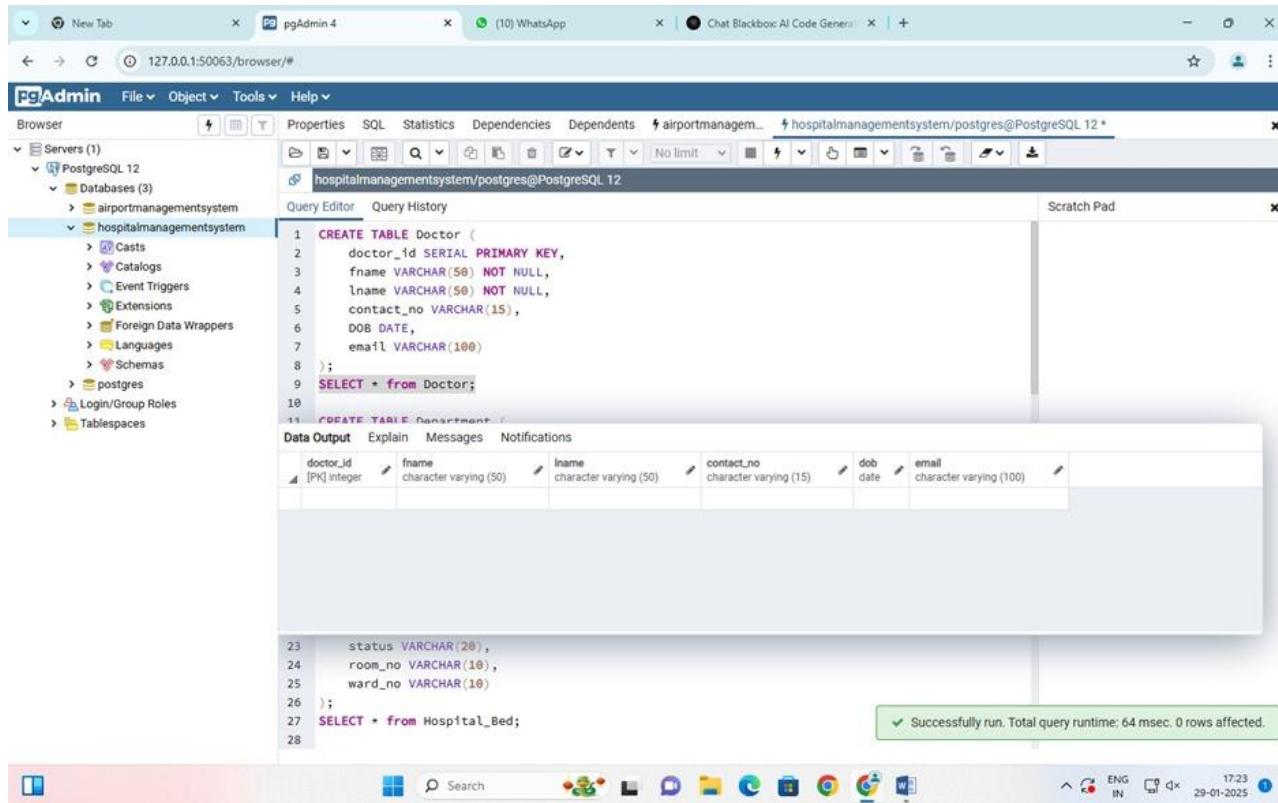
<ColumnName><Datatype>(<size>)

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

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

Problem Statement

The problem revolves around designing a database for a Hospital Management System. It manages patient details, doctor information, department allocations, hospital bed assignments, appointment scheduling, medical records, prescriptions, and billing details, while maintaining essential constraints, specializations, generalizations, and relationships to ensure accurate and efficient management of hospital operations.



The screenshot shows the pgAdmin 4 interface connected to a PostgreSQL 12 database named 'hospitalmanagementsystem'. In the 'Query Editor' tab, the following SQL code is run:

```

1 CREATE TABLE Doctor (
2     doctor_id SERIAL PRIMARY KEY,
3     fname VARCHAR(50) NOT NULL,
4     lname VARCHAR(50) NOT NULL,
5     contact_no VARCHAR(15),
6     DOB DATE,
7     email VARCHAR(100)
8 );
9 SELECT * from Doctor;
10
11 CREATE TABLE Department (
12     department_id SERIAL PRIMARY KEY,
13     name VARCHAR(50) NOT NULL,
14     specialization VARCHAR(50),
15     status VARCHAR(20),
16     room_no VARCHAR(10),
17     ward_no VARCHAR(10)
18 );
19 SELECT * from Department;
20
21 CREATE TABLE Hospital_Bed (
22     bed_id SERIAL PRIMARY KEY,
23     status VARCHAR(20),
24     room_no VARCHAR(10),
25     ward_no VARCHAR(10)
26 );
27 SELECT * from Hospital_Bed;
28
  
```

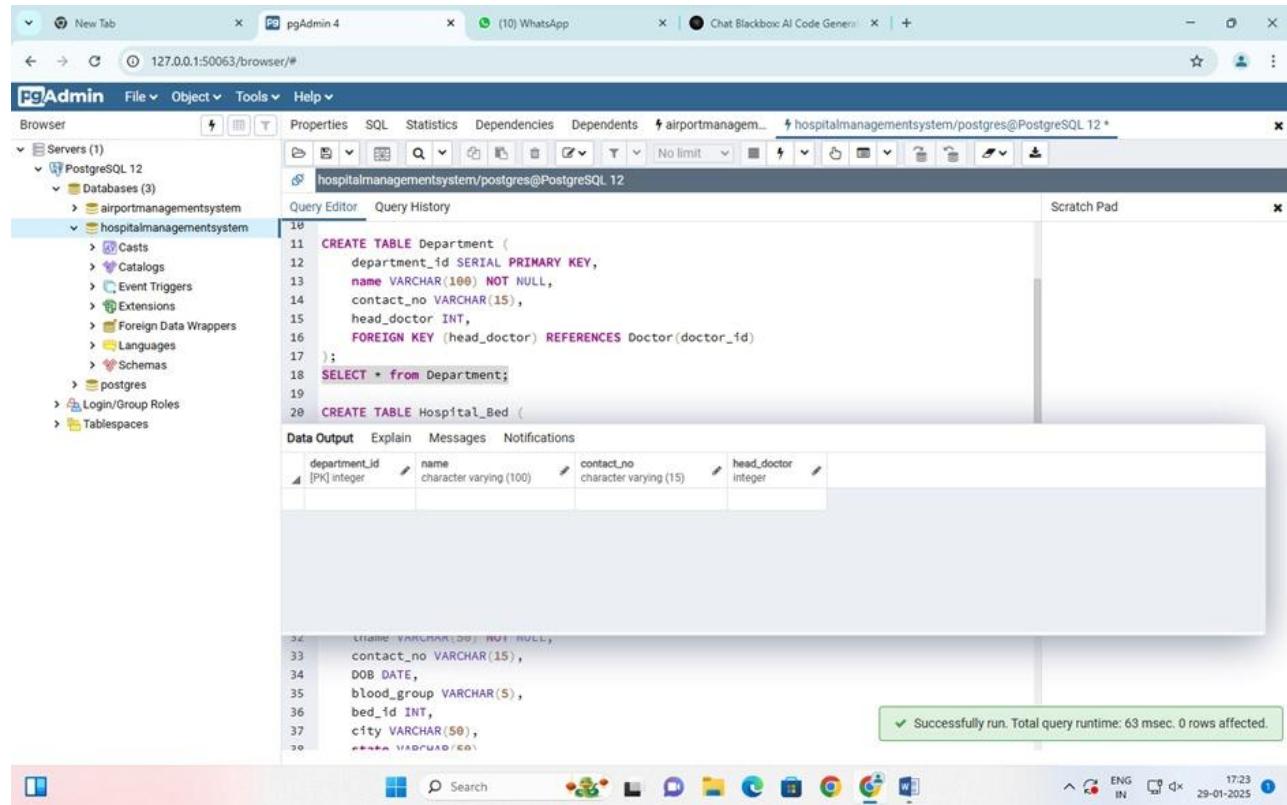
The 'Data Output' tab displays the schema of the 'Doctor' table:

doctor_id	fname	lname	contact_no	dob	email
[PK] Integer	character varying (50)	character varying (50)	character varying (15)	date	character varying (100)

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 64 msec. 0 rows affected."

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025



The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1)', 'PostgreSQL 12' is selected, showing 'Databases (3)': 'airportmanagementsystem', 'hospitalmanagementsystem', and 'postgres'. The 'hospitalmanagementsystem' database is currently selected. The 'Query Editor' tab contains the following SQL code:

```

10
11 CREATE TABLE Department (
12   department_id SERIAL PRIMARY KEY,
13   name VARCHAR(100) NOT NULL,
14   contact_no VARCHAR(15),
15   head_doctor INT,
16   FOREIGN KEY (head_doctor) REFERENCES Doctor(doctor_id)
17 );
18 SELECT * from Department;
19
20 CREATE TABLE Hospital_Bed (
21   bed_id integer [PK] primary key,
22   patient_name VARCHAR(50) NOT NULL,
23   contact_no VARCHAR(15),
24   DOB DATE,
25   blood_group VARCHAR(5),
26   bed_id INT,
27   city VARCHAR(50),
28   status VARCHAR(50)
29

```

The 'Data Output' tab shows the schema for the 'Hospital_Bed' table:

department_id	name	contact_no	head_doctor

A green message bar at the bottom right indicates: 'Successfully run. Total query runtime: 63 msec. 0 rows affected.'

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025

The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** PostgreSQL 12
- Databases:** hospitalmanagementsystem
- Table Structure:** Hospital_Bed (bed_id, type, status, room_no, ward_no)
- Table Structure:** Patient (patient_id, landmark, bed_id, status, room_no, ward_no)
- Table Structure:** Appointment (appointment_id, patient_id, bed_id, date, time, duration, status)

The SQL code in the Query Editor is as follows:

```
CREATE TABLE Hospital_Bed (
    bed_id SERIAL PRIMARY KEY,
    type VARCHAR(50),
    status VARCHAR(20),
    room_no VARCHAR(10),
    ward_no VARCHAR(10)
);
SELECT * from Hospital_Bed;

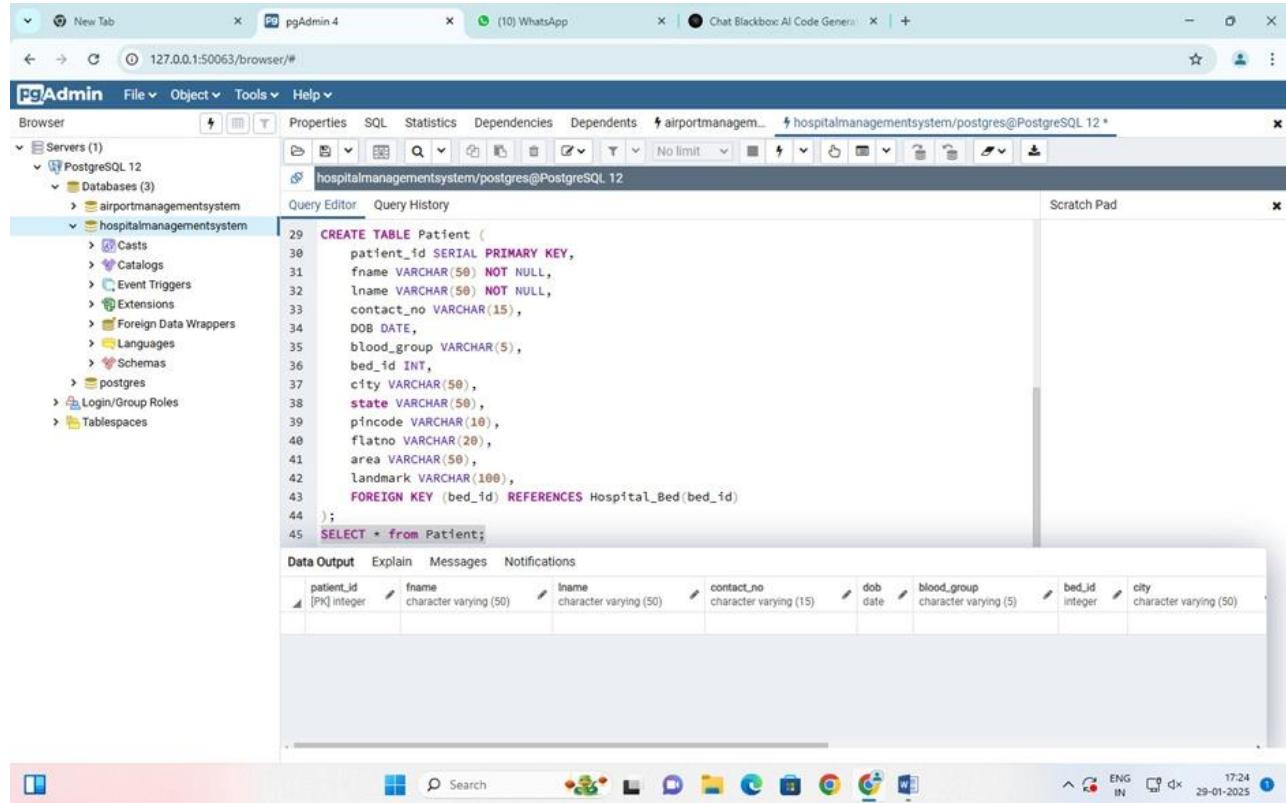
CREATE TABLE Patient (
    patient_id SERIAL PRIMARY KEY,
    landmark VARCHAR(100),
    FOREIGN KEY (bed_id) REFERENCES Hospital_Bed(bed_id)
);
SELECT * from Patient;

CREATE TABLE Appointment (
    appointment_id SERIAL PRIMARY KEY,
    patient_id INTEGER,
    bed_id INTEGER,
    date DATE,
    time TIME,
    duration INTERVAL,
    status VARCHAR(20)
);
```

A message at the bottom right indicates: "Successfully run. Total query runtime: 67 msec. 0 rows affected."

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025



```

CREATE TABLE Patient (
  patient_id SERIAL PRIMARY KEY,
  fname VARCHAR(50) NOT NULL,
  lname VARCHAR(50) NOT NULL,
  contact_no VARCHAR(15),
  dob DATE,
  blood_group VARCHAR(5),
  bed_id INT,
  city VARCHAR(50),
  state VARCHAR(50),
  pincode VARCHAR(10),
  flatno VARCHAR(20),
  area VARCHAR(50),
  landmark VARCHAR(100),
  FOREIGN KEY (bed_id) REFERENCES Hospital_Bed(bed_id)
);
SELECT * from Patient;
  
```

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows 'PostgreSQL 12' with databases 'airportmanagementsystem' and 'hospitalmanagementsystem'. The 'hospitalmanagementsystem' database is selected. In the center, the 'Query Editor' tab contains the SQL code for creating a 'Patient' table. The table has columns for patient_id (primary key, SERIAL), fname (VARCHAR(50)), lname (VARCHAR(50)), contact_no (VARCHAR(15)), dob (DATE), blood_group (VARCHAR(5)), bed_id (INT), city (VARCHAR(50)), state (VARCHAR(50)), pincode (VARCHAR(10)), flatno (VARCHAR(20)), area (VARCHAR(50)), and landmark (VARCHAR(100)). It includes a foreign key constraint for bed_id referencing the 'Hospital_Bed' table. Below the code, a 'Data Output' tab shows the table structure with columns: patient_id [PK] integer, fname character varying (50), lname character varying (50), contact_no character varying (15), dob date, blood_group character varying (5), bed_id integer, and city character varying (50). The bottom status bar shows the date as 29-01-2025 and time as 17:24.

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025

The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** PostgreSQL 12
- Databases:** hospitalmanagementsystem (selected)
- Query Editor:** Displays the SQL code for creating three tables: Hospital_Bed, Appointment, and Medical_Record.
- Table Structure:** Shows the columns and data types for the Appointment table.
- Status:** A green message at the bottom right indicates "Successfully run. Total query runtime: 65 msec. 0 rows affected."

```
23    FOREIGN KEY (bed_id) REFERENCES Hospital_Bed(bed_id)
24  );
25  SELECT * from Patient;
26
27 CREATE TABLE Appointment (
28   appointment_id SERIAL PRIMARY KEY,
29   patient_id INT,
30   doctor_id INT,
31   date TIMESTAMP,
32   status VARCHAR(20),
33   fees DECIMAL(10,2),
34   FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
35   FOREIGN KEY (doctor_id) REFERENCES Doctor(doctor_id)
36 );
37  SELECT * from Appointment;
38
39 CREATE TABLE Medical_Record (
40   record_id SERIAL PRIMARY KEY,
```

appointment_id	patient_id	doctor_id	date	status	fees
[PK] integer	integer	integer	timestamp without time zone	character varying (20)	numeric (10,2)

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1)', 'PostgreSQL 12' is selected, showing 'Databases (3)': 'airportmanagementsystem', 'hospitalmanagementsystem', and 'postgres'. The 'hospitalmanagementsystem' database is currently selected. The 'Query Editor' tab contains the following SQL code:

```

57 SELECT * from Appointment;
58
59 CREATE TABLE Medical_Record (
60   record_id SERIAL PRIMARY KEY,
61   patient_id INT,
62   diagnosis TEXT,
63   treatment TEXT,
64   date DATE,
65   FOREIGN KEY (patient_id) REFERENCES Patient(patient_id)
66 );
67 SELECT * from Medical_Record;
68
69 CREATE TABLE Bill (
70   bill_id SERIAL PRIMARY KEY,
71   appointment_id INT,
72   date DATE,
73   amount DECIMAL(10,2),
74   status VARCHAR(20).

```

The 'Data Output' tab shows the structure of the 'Medical_Record' table:

record_id	patient_id	diagnosis	treatment	date
[PK] integer	integer	text	text	date

Two green success messages are visible at the bottom right of the query editor:

- Successfully run. Total query runtime: 66 ms
- Successfully run. Total query runtime: 63 ms

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025

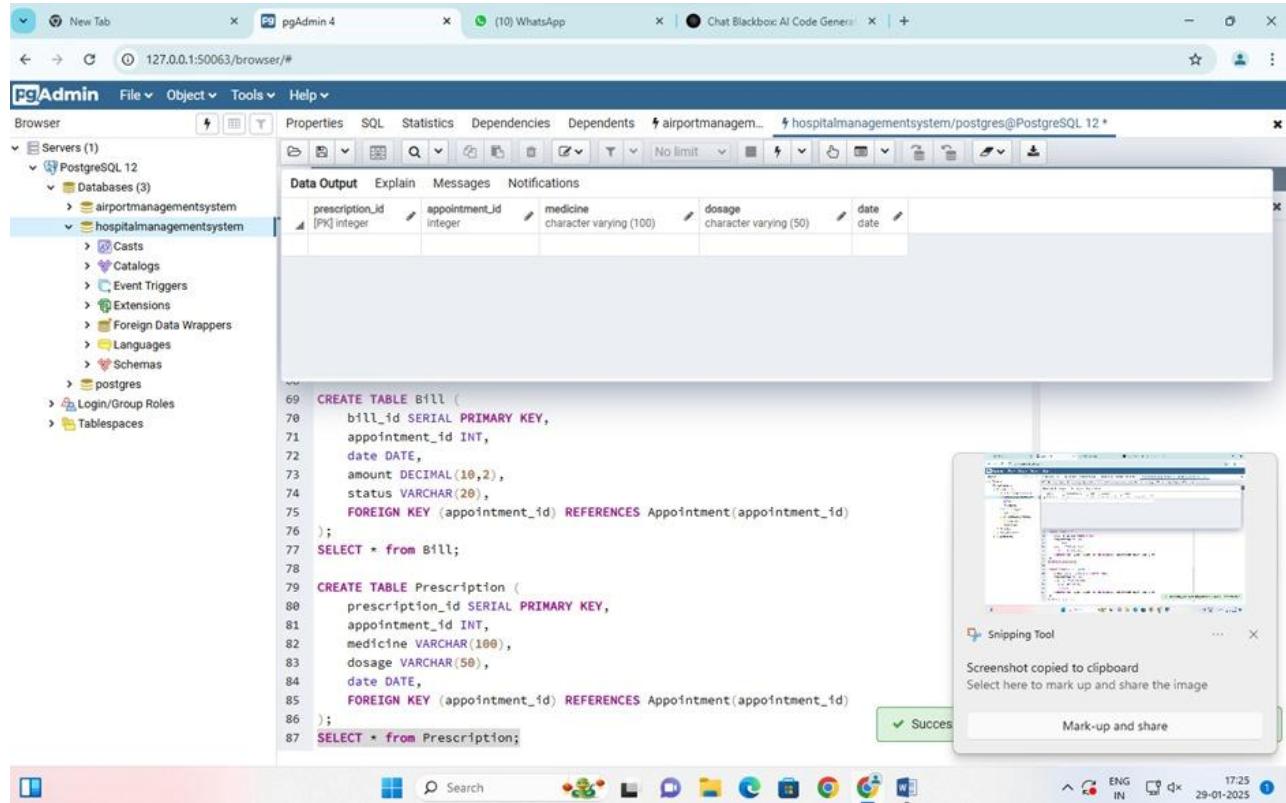
```

69 CREATE TABLE Bill (
70   bill_id SERIAL PRIMARY KEY,
71   appointment_id INT,
72   date DATE,
73   amount DECIMAL(10,2),
74   status VARCHAR(20),
75   FOREIGN KEY (appointment_id) REFERENCES Appointment(appointment_id)
76 );
77 SELECT * from Bill;
78
79 CREATE TABLE Prescription (
80   prescription_id SERIAL PRIMARY KEY,
81   appointment_id INT,
82   medicine VARCHAR(100),
83   dosage VARCHAR(50),
84   date DATE,
85   FOREIGN KEY (appointment_id) REFERENCES Appointment(appointment_id)
86 );
87 SELECT * from Prescription;
  
```

Successfully run. Total query runtime: 68 msec. 0 rows affected.

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025



The screenshot shows the pgAdmin 4 interface. On the left, the object browser displays a tree structure of databases, including 'PostgreSQL 12' and 'hospitalmanagementsystem'. The 'hospitalmanagementsystem' database is selected, showing its tables: 'Casts', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Schemas', and 'postgres'. The main pane shows the 'Data Output' tab for a table named 'prescription'. The table has columns: prescription_id [PK] integer, appointment_id integer, medicine character varying (100), dosage character varying (50), and date date. Below the table, the SQL query for creating the 'Bill' table is displayed:

```

69 CREATE TABLE Bill (
70   bill_id SERIAL PRIMARY KEY,
71   appointment_id INT,
72   date DATE,
73   amount DECIMAL(10,2),
74   status VARCHAR(20),
75   FOREIGN KEY (appointment_id) REFERENCES Appointment(appointment_id)
76 );
77 SELECT * from Bill;
78
79 CREATE TABLE Prescription (
80   prescription_id SERIAL PRIMARY KEY,
81   appointment_id INT,
82   medicine VARCHAR(100),
83   dosage VARCHAR(50),
84   date DATE,
85   FOREIGN KEY (appointment_id) REFERENCES Appointment(appointment_id)
86 );
87 SELECT * from Prescription;

```

A context menu is open over the SQL editor area, with options like 'Snipping Tool', 'Screenshot copied to clipboard', 'Select here to mark up and share the image', and 'Mark-up and share'.

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025

Post Lab Questions:

1. Explain in brief the following terms:

- a. **Database** - A database is an organized collection of data, so that it can be easily accessed and managed. You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.
- b. **Types of databases**
 - Centralized Database
 - Distributed Database
 - NoSQL Database
 - Cloud Database
 - Relational Database
 - Network Database
 - Object-oriented Database
 - Hierarchical Database
- c. **SQL Data Types**
 - Numeric Data Types
 - Character and String Data Types
 - Date and Time Data Types
 - Binary Data Types
 - Boolean Data Types
 - Special Data Types
- d. **Foreign key** - In the relational databases, a foreign key is a field or a column that is used to establish a link between two tables. In simple words you can say that, a foreign key in one table used to point primary key in another table.



2. What are the commands to:

a. Delete an entire table

Ans. `DROP TABLE table_name;`

b. To view a database

Ans. `SHOW DATABASES;`

`USE database_name;`

`SHOW TABLES;`

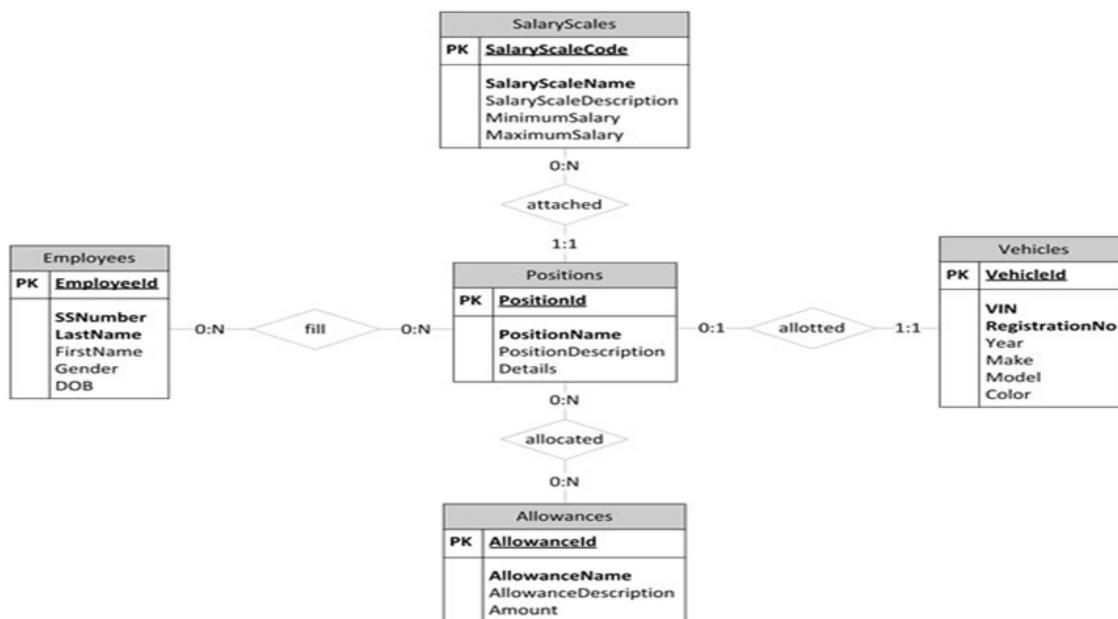
c. To select & view all the columns

Ans. `SELECT * FROM table_name;`

3. 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.





Code-

```
CREATE TABLE SalaryScales (
    SalaryScaleCode INT PRIMARY KEY,
    SalaryScaleName VARCHAR(100) NOT NULL,
    SalaryScaleDescription TEXT,
    MinimumSalary DECIMAL(10, 2) NOT NULL,
    MaximumSalary DECIMAL(10, 2) NOT NULL,
    CHECK (MinimumSalary <= MaximumSalary)
);
```

```
CREATE TABLE Employees (
    EmployeeId INT PRIMARY KEY,
    SSNumber VARCHAR(15) UNIQUE NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    FirstName VARCHAR(50) NOT NULL,
    Gender CHAR(1) CHECK (Gender IN ('M', 'F', 'O')),
    DOB DATE NOT NULL
);
```

```
CREATE TABLE Positions (
    PositionId INT PRIMARY KEY,
    PositionName VARCHAR(100) NOT NULL,
```

Department of Computer Engineering



```
PositionDescription TEXT,  
Details TEXT,  
SalaryScaleCode INT UNIQUE,  
FOREIGN KEY (SalaryScaleCode) REFERENCES SalaryScales(SalaryScaleCode)  
);
```

```
CREATE TABLE Vehicles (  
    VehicleId INT PRIMARY KEY,  
    VIN VARCHAR(20) UNIQUE NOT NULL,  
    RegistrationNo VARCHAR(20) UNIQUE NOT NULL,  
    Year_R INT NOT NULL,  
    Make VARCHAR(50) NOT NULL,  
    Model VARCHAR(50) NOT NULL,  
    Color VARCHAR(30) NOT NULL  
);
```

```
CREATE TABLE Allowances (  
    AllowanceId INT PRIMARY KEY,  
    AllowanceName VARCHAR(100) NOT NULL,  
    AllowanceDescription TEXT,  
    Amount DECIMAL(10, 2) NOT NULL  
);
```

```
CREATE TABLE Employee_Positions (  
    EmployeeId INT,  
    Department of Computer Engineering
```



```
PositionId INT,  
PRIMARY KEY (EmployeeId, PositionId),  
FOREIGN KEY (EmployeeId) REFERENCES Employees(EmployeeId),  
FOREIGN KEY (PositionId) REFERENCES Positions(PositionId)  
);
```

```
CREATE TABLE Position_Allowances (  
PositionId INT,  
AllowanceId INT,  
PRIMARY KEY (PositionId, AllowanceId),  
FOREIGN KEY (PositionId) REFERENCES Positions(PositionId),  
FOREIGN KEY (AllowanceId) REFERENCES Allowances(AllowanceId)  
);
```

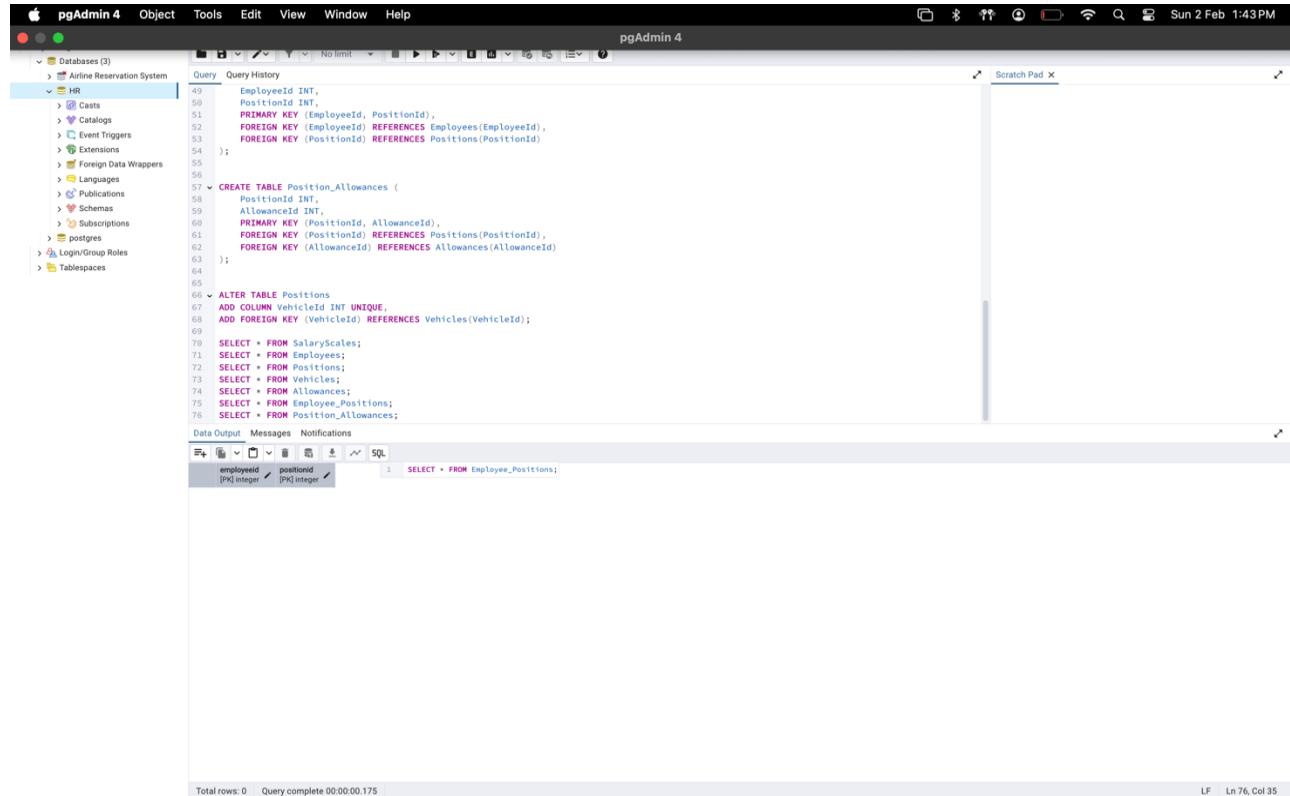
```
ALTER TABLE Positions  
ADD COLUMN VehicleId INT UNIQUE,  
ADD FOREIGN KEY (VehicleId) REFERENCES Vehicles(VehicleId);
```

```
SELECT * FROM SalaryScales;  
SELECT * FROM Employees;  
SELECT * FROM Positions;  
SELECT * FROM Vehicles;  
SELECT * FROM Allowances;
```

Department of Computer Engineering

SELECT * FROM Employee_Positions;

SELECT * FROM Position_Allowances;



```

pgAdmin 4 Object Tools Edit View Window Help
pgAdmin 4
Databases (3) Airline Reservation System Query Query History
  > HR
    > Casts
    > Catalogs
    > Event Triggers
    > Extensions
    > Foreign Data Wrappers
    > Languages
    > Publications
    > Schemas
    > Subscriptions
    > postgres
    > Login/Group Roles
    > Tablespaces
  49   EmployeeId INT;
  50   PositionId INT;
  51   PRIMARY KEY (EmployeeId, PositionId),
  52   FOREIGN KEY (EmployeeId) REFERENCES Employees(EmployeeId),
  53   FOREIGN KEY (PositionId) REFERENCES Positions(PositionId)
  54  );
  55
  56  ✓ CREATE TABLE Position_Allowances (
  57   PositionId INT,
  58   AllowanceId INT,
  59   PRIMARY KEY (PositionId, AllowanceId),
  60   FOREIGN KEY (PositionId) REFERENCES Positions(PositionId),
  61   FOREIGN KEY (AllowanceId) REFERENCES Allowances(AllowanceId)
  62  );
  63
  64
  65
  66 ✓ ALTER TABLE Positions
  67 ADD COLUMN VehicleId INT UNIQUE;
  68 ADD FOREIGN KEY (VehicleId) REFERENCES Vehicles(VehicleId);
  69
  70 SELECT * FROM SalaryScales;
  71 SELECT * FROM Employees;
  72 SELECT * FROM Positions;
  73 SELECT * FROM Vehicles;
  74 SELECT * FROM Allowances;
  75 SELECT * FROM Employee_Positions;
  76 SELECT * FROM Position_Allowances;
  
```

Total rows: 0 Query complete 00:00:00.175 LF Ln 76, Col 35

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure for 'Airlne Reservation System' under the 'Databases' section. The 'HR' schema is selected, showing tables like 'Casts', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', 'Schemas', 'Subscriptions', 'postgres', 'Login/Group Roles', and 'Tablespaces'. The 'Query' tab in the top right is active, displaying a multi-line SQL script. The script includes the creation of the 'Position_Allowances' table with columns 'PositionId' (INT, PK), 'AllowanceId' (INT, PK), and foreign keys linking to 'Employees' (EmployeeId), 'Positions' (PositionId), and 'Allowances' (AllowanceId). It also includes an 'ALTER TABLE Positions' command adding a column 'VehicleId' with a unique constraint and a foreign key reference to 'Vehicles' (VehicleId). Below the script, the 'Data Output' tab is visible, showing a table structure for 'Position' with columns 'PositionId' (PK integer) and 'PositionName' (character varying(100)). A 'SQL' tab is also present. At the bottom, status bars show 'Total rows: 0' and 'Query complete 00:00:00.128'.

Department of Computer Engineering

RDBMS –Sem-IV- Jan –April 2025



Conclusion:

SQL DDL is essential for defining and managing database structures. Commands like CREATE, ALTER, and DROP help create, modify, and delete tables while maintaining data integrity through constraints.