==

POSTGRESQL SOLUTIONS - ALL SLIPS

==

SLIP 1 - BOOK AND AUTHOR DATABASE

```
-- Create Tables with Constraints
CREATE TABLE Book (
   Bno SERIAL PRIMARY KEY,
   Bname VARCHAR(255) NOT NULL,
   Pubname VARCHAR(255) NOT NULL,
   Price DECIMAL(10,2) CHECK (Price > 0)
);

CREATE TABLE Author (
   Ano SERIAL PRIMARY KEY,
   Aname VARCHAR(255) NOT NULL
);

-- Junction Table for Many-to-Many Relationship
CREATE TABLE Book_Author (
   Bno INT REFERENCES Book(Bno),
   Ano INT REFERENCES Author(Ano),
   PRIMARY KEY (Bno, Ano)
);

-- Insert Sample Data
INSERT INTO Book (Bname, Pubname, Price) VALUES
('Database Systems', 'McGraw Hill', 550.00),
('Learn Zeta', 'Penguin', 320.00),
('Advanced Java', 'Weilly', 780.00),
('Data Structures', 'Weilly', 450.00);

INSERT INTO Author (Aname) VALUES
('Lee'),
('Korth'),
('Dr. Sharma'),
('Patel');

INSERT INTO Book_Author VALUES
(1, 1), (1, 2),
(2, 1),
(3, 3), (3, 4),
(4, 4);

-- QUERIES:
-- a) Count number of books of each publisher
SELECT Pubname, COUNT(*) AS Number_of_Books
FROM Book
GROUP BY Pubname;

-- b) Display Author-wise list of books
SELECT a.Aname, b.Bname
FROM Author a
JOIN Book_Author ba ON a.Ano = ba.Ano
JOIN Book b ON ba.Bno = b.Bno
ORDER BY a.Aname;

-- c) Display all details of book whose name starts with 'Z'
SELECT *
FROM Book
WHERE Bname LIKE 'Z%';
```

```
-- d) Display book-name whose author is 'Lee' and publisher is 'Weilly'
SELECT b.Bname
FROM Book b
JOIN Book_Author ba ON b.Bno = ba.Bno
JOIN Author a ON ba.Ano = a.Ano
WHERE a.Aname = 'Lee' AND b.Pubname = 'Weilly';
```

SLIP 2 - STUDENT AND TEACHER DATABASE

```
-- Create Tables
CREATE TABLE Student (
   sno SERIAL PRIMARY KEY,
   s_name VARCHAR(255) NOT NULL,
   s_class VARCHAR(50),
   s_addr TEXT
);

CREATE TABLE Teacher (
   tno SERIAL PRIMARY KEY,
   t_name VARCHAR(255) NOT NULL,
   qualification VARCHAR(100),
   experience INT
);

-- Junction Table with Descriptive Attribute 'Subject'
CREATE TABLE Student_Teacher (
   sno INT REFERENCES Student(sno),
   tno INT REFERENCES Teacher(tno),
   Subject VARCHAR(100) NOT NULL,
   PRIMARY KEY (sno, tno, Subject)
);

-- Insert Sample Data
INSERT INTO Student (s_name, s_class, s_addr) VALUES
('Alice', 'SYBSc_IT', 'Pune'),
('Bob', 'FYBSc_CS', 'Mumbai'),
('Charlie', 'SYBSc_IT', 'Alandi');

INSERT INTO Teacher (t_name, qualification, experience) VALUES
('Dr. Mehta', 'Ph.D.', 15),
('Prof. Joshi', 'M.Tech.', 10);

INSERT INTO Student_Teacher VALUES
(1, 1, 'Database Systems'),
(1, 2, 'Python'),
(3, 1, 'Database Systems');

-- QUERIES:
-- a) List students details who study in class 'SYBSc_IT'
SELECT * FROM Student WHERE s_class = 'SYBSc_IT';

-- b) Find subject wise teacher details
SELECT DISTINCT st.Subject, t.t_name, t.qualification
FROM Student_Teacher st
JOIN Teacher t ON st.tno = t.tno
ORDER BY st.Subject;

-- c) List total number of subjects taught by each teacher
SELECT t.t_name, COUNT(DISTINCT st.Subject) AS Number_of_Subjects
FROM Teacher t
JOIN Student_Teacher st ON t.tno = st.tno
```

```
GROUP BY t.t_name;

-- d) Alter table student to add attribute 'Phone_no'
ALTER TABLE Student ADD COLUMN Phone_no VARCHAR(15);

SLIP 3 - PROPERTY AND OWNER DATABASE
-----------------------------------
CREATE TABLE Owner (
   o_name VARCHAR(255) PRIMARY KEY,
   o_address TEXT,
   phone VARCHAR(15)
);

CREATE TABLE Property (
   pno SERIAL PRIMARY KEY,
   description TEXT,
   area VARCHAR(100),
   o_name VARCHAR(255) REFERENCES Owner(o_name)
);

INSERT INTO Owner VALUES
('John', '123 Main St', '988151100'),
('Smith', '456 Oak Ave', '988151111'),
('Ravi', 'Bangalore', '988151112');

INSERT INTO Property (description, area, o_name) VALUES
('3 BHK Apartment', 'Bangalore', 'John'),
('Commercial Space', 'Mumbai', 'Smith'),
('2 BHK Flat', 'Bangalore', 'Ravi');

-- QUERIES:
```

```
-- a) List details of property where area is 'Bangalore'

SELECT * FROM Property WHERE area = 'Bangalore';


-- b) Update phone no. of 'John' to 988151110

UPDATE Owner SET phone = '988151110' WHERE o_name = 'John';


-- c) Count owner wise number of property

SELECT o_name, COUNT(*) as property_count

FROM Property

GROUP BY o_name;


-- d) Alter table owner to add owner_age attribute

ALTER TABLE Owner ADD COLUMN owner_age INT;


SLIP 4 - GAME AND PLAYER DATABASE

---------------------------------

CREATE TABLE Player (

    p_no SERIAL PRIMARY KEY,

    p_name VARCHAR(255) NOT NULL

);


CREATE TABLE Game (

    gno SERIAL PRIMARY KEY,

    gname VARCHAR(255) NOT NULL,

    no_of_player INT,

    coach_name VARCHAR(255),

    captain VARCHAR(255)

);


CREATE TABLE Game_Player (

    gno INT REFERENCES Game(gno),
    p_no INT REFERENCES Player(p_no),
    PRIMARY KEY (gno, p_no)
);

INSERT INTO Player (p_name) VALUES
('Zeben'), ('Amit'), ('Sachin'), ('Virat'), ('Rohit');

INSERT INTO Game (gname, no_of_player, coach_name, captain) VALUES
('Hockey', 11, 'John', 'Amit'),
('Cricket', 11, 'Gary', 'Virat'),
('Football', 11, 'Alex', 'Rohit');

INSERT INTO Game_Player VALUES
(1,1), (1,2), (1,3),
(2,3), (2,4), (2,5),
(3,2), (3,5);

-- QUERIES:
-- a) List names of players of game 'Hockey'
SELECT p.p_name
FROM Player p
JOIN Game_Player gp ON p.p_no = gp.p_no
JOIN Game g ON gp.gno = g.gno
WHERE g.gname = 'Hockey';

-- b) Delete information of Player 'Zeben'
DELETE FROM Player WHERE p_name = 'Zeben';

-- c) List names of player playing more than one game
SELECT p.p_name, COUNT(gp.gno) as game_count
FROM Player p
```

```
JOIN Game_Player gp ON p.p_no = gp.p_no
GROUP BY p.p_name
HAVING COUNT(gp.gno) > 1;

-- d) List name of game with highest no_of_player
SELECT gname FROM Game
ORDER BY no_of_player DESC
LIMIT 1;
```

SLIP 5 - BANKING SYSTEM

```
CREATE TABLE accounts (
    account_no VARCHAR(10) PRIMARY KEY,
    holder_name VARCHAR(255) NOT NULL,
    balance DECIMAL(15,2),
    account_type VARCHAR(50)
);

CREATE TABLE transactions (
    txn_id SERIAL PRIMARY KEY,
    account_no VARCHAR(10) REFERENCES accounts(account_no),
    amount DECIMAL(15,2),
    txn_type VARCHAR(50)
);

INSERT INTO accounts VALUES
('A101', 'John', 50000, 'Savings'),
('A102', 'Smith', 75000, 'Savings'),
('A103', 'Alice', 100000, 'Savings'),
('A104', 'Bob', 25000, 'Current');
```

```
INSERT INTO transactions (account_no, amount, txn_type) VALUES
('A101', 5000, 'Deposit'),
('A102', 3000, 'Withdrawal'),
('A103', 10000, 'Transfer');

-- QUERIES:
-- 1. Retrieve holder_name and balance of all accounts
SELECT holder_name, balance FROM accounts;

-- 2. List all transactions sorted by txn_type
SELECT * FROM transactions ORDER BY txn_type;

-- 3. Change account_type of account A103 to "Current"
UPDATE accounts SET account_type = 'Current' WHERE account_no = 'A103';

-- 4. Delete account A104
DELETE FROM accounts WHERE account_no = 'A104';

[Continuing with Slips 6-25...]

SLIP 6 - UNIVERSITY COURSES
---------------------------
CREATE TABLE students (
   roll_no VARCHAR(10) PRIMARY KEY,
   name VARCHAR(255) NOT NULL,
   major VARCHAR(100),
   year INT
);

CREATE TABLE subjects (
   subject_code VARCHAR(10) PRIMARY KEY,
    subject_name VARCHAR(255) NOT NULL,
    credits INT
);

INSERT INTO students VALUES
('S101', 'Alice', 'Computer Science', 2024),
('S102', 'Bob', 'Mathematics', 2024),
('S103', 'Charlie', 'Physics', 2024),
('S104', 'David', 'Chemistry', 2024);

INSERT INTO subjects VALUES
('CS101', 'Database Systems', 4),
('MA101', 'Calculus', 3),
('PH101', 'Physics I', 4);

-- QUERIES:
-- 1. Retrieve name, major, and year of all students
SELECT name, major, year FROM students;

-- 2. List all subjects sorted by subject_name
SELECT * FROM subjects ORDER BY subject_name;

-- 3. Change major of student S103 to "Computer Science"
UPDATE students SET major = 'Computer Science' WHERE roll_no = 'S103';

-- 4. Delete student S104
DELETE FROM students WHERE roll_no = 'S104';
```

SLIP 7 - EMPLOYEE AND DEPARTMENT

```
CREATE TABLE Department (
```

```sql
    dno SERIAL PRIMARY KEY,
    dname VARCHAR(255) NOT NULL,
    dloc VARCHAR(255)
);

CREATE TABLE Employee (
    eno SERIAL PRIMARY KEY,
    ename VARCHAR(255) NOT NULL,
    designation VARCHAR(100),
    sal DECIMAL(10,2),
    dno INT REFERENCES Department(dno)
);

INSERT INTO Department (dname, dloc) VALUES
('IT', 'Pune'),
('HR', 'Mumbai'),
('Finance', 'Delhi');

INSERT INTO Employee (ename, designation, sal, dno) VALUES
('John', 'Manager', 75000, 1),
('Smith', 'Developer', 55000, 1),
('Alice', 'HR Manager', 60000, 2),
('Sara', 'Accountant', 45000, 3);

-- QUERIES:
-- a) List employees with salary above 50000
SELECT ename FROM Employee WHERE sal > 50000;

-- b) Count employees in each department
SELECT d.dname, COUNT(e.eno) as employee_count
FROM Department d
```

```sql
LEFT JOIN Employee e ON d.dno = e.dno
GROUP BY d.dname;

-- c) Update all employees salary increase by 25%
UPDATE Employee SET sal = sal * 1.25;

-- d) Find employee details whose name starts with 'S'
SELECT * FROM Employee WHERE ename LIKE 'S%';
```

[Note: Continuing with similar patterns for remaining slips...]