

Here is a **structured SQL practice setup** to help you master the queries you've listed. I will:

1. **Create 4 main tables:** car, employee, department, customer, product, order, student.
 2. **Provide sample schema** with CREATE TABLE statements and example rows.
 3. **Group your practice questions** under each table/topic from **basic to advanced**.
-

Step 1: Create Tables

-- CAR Table

```
CREATE TABLE car (  
    id SERIAL PRIMARY KEY,  
    model VARCHAR(50),  
    color VARCHAR(20),  
    price NUMERIC(10,2)  
);
```

-- DEPARTMENT Table

```
CREATE TABLE department (  
    id SERIAL PRIMARY KEY,  
    dept_name VARCHAR(50)  
);
```

-- EMPLOYEE Table

```
CREATE TABLE employee (  
    id SERIAL PRIMARY KEY,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50),  
    salary NUMERIC(10,2),  
    dept_id INT REFERENCES department(id),  
    manager_id INT,  
    join_date DATE  
);
```

-- CUSTOMER Table

```
CREATE TABLE customer (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(50)  
);
```

-- PRODUCT Table

```
CREATE TABLE product (  
    id SERIAL PRIMARY KEY,  
    product_name VARCHAR(50),  
    price NUMERIC(10,2)  
);
```

-- ORDERS Table

```
CREATE TABLE orders (  
    id SERIAL PRIMARY KEY,  
    customer_id INT REFERENCES customer(id),  
    product_id INT REFERENCES product(id),  
    quantity INT,  
    order_date DATE  
);
```

-- STUDENT Table

```
CREATE TABLE student (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(50),  
    age INT,  
    dept VARCHAR(50)  
);
```

Here are **sample INSERT statements** with realistic data for each of the tables you created. These will help you practice all your SQL queries, including joins, views, functions, triggers, and more.

car Table

```
INSERT INTO car (model, color, price) VALUES
('Swift', 'Red', 650000.00),
('Alto', 'Blue', 450000.00),
('Creta', 'White', 1050000.00),
('Fortuner', 'Black', 3500000.00),
('Polo', 'Red', 750000.00),
('Indica', 'Yellow', 300000.00),
('i20', 'White', 800000.00),
('Harrier', 'Black', 2000000.00),
('Baleno', 'Silver', 700000.00),
('Jazz', 'Green', 720000.00);
```

department Table

```
INSERT INTO department (dept_name) VALUES
('Engineering'),
('HR'),
('Marketing'),
('Finance'),
('Support');
```

employee Table

```
INSERT INTO employee (first_name, last_name, salary, dept_id, manager_id, join_date) VALUES
('David', 'Lee', 95000, 1, NULL, '2021-01-15'),
('Daniel', 'Craig', 70000, 1, 1, '2023-06-10'),
('Emma', 'Watson', 80000, 2, 1, '2022-02-20'),
('Olivia', 'Brown', 75000, 2, 3, '2023-01-01'),
('James', 'Bond', 90000, 3, 1, '2021-09-30'),
```

```
('Daisy', 'Ridley', 65000, 3, 5, '2024-03-10'),  
('Ethan', 'Hunt', 100000, 1, 1, '2020-11-11'),  
('Alice', 'Wong', 72000, 4, 7, '2023-05-20'),  
('Mark', 'Johnson', 85000, 5, 7, '2022-07-30'),  
('John', 'Doe', 78000, 5, 7, '2021-12-25');
```

customer Table

```
INSERT INTO customer (name) VALUES  
  
('Riya'),  
('Kabir'),  
('Anjali'),  
('Vikram'),  
('Sanya'),  
('Raj'),  
('Nina');
```

product Table

```
INSERT INTO product (product_name, price) VALUES  
  
('Laptop', 55000),  
('Mouse', 500),  
('Keyboard', 1500),  
('Monitor', 7000),  
('Phone', 30000),  
('Tablet', 25000),  
('Printer', 12000),  
('Camera', 45000);
```

orders Table

```
INSERT INTO orders (customer_id, product_id, quantity, order_date) VALUES  
  
(1, 1, 1, '2023-06-01'),  
(1, 2, 2, '2023-06-01'),
```

```
(2, 5, 1, '2023-06-02'),  
(2, 6, 1, '2023-06-02'),  
(3, 4, 1, '2023-06-03'),  
(4, 7, 1, '2023-06-04'),  
(5, 1, 1, '2023-06-05'),  
(6, 3, 2, '2023-06-06');  
-- Customer 7 (Nina) placed no orders
```

student Table

INSERT INTO student (name, age, dept) VALUES

```
('Ravi', 21, 'Engineering'),  
( 'Meera', 22, 'Marketing'),  
( 'Arjun', 21, 'Engineering'),  
( 'Tina', 23, 'HR'),  
( 'Karan', 22, 'Finance'),  
( 'Simran', 22, 'Marketing'),  
( 'Ali', 21, 'Engineering'),  
( 'Neha', 24, 'HR'),  
( 'Dev', 25, 'Law'),  
( 'Rohan', 21, 'Medicine');
```

Notes







- **Manager IDs** in employee point to another employee (manager_id is nullable).
- **Foreign keys:**
 - employee.dept_id → department.id
 - orders.customer_id → customer.id
 - orders.product_id → product.id
- You can now easily test:
 - View creation
 - Join queries
 - Aggregates, groupings

- Subqueries
 - Window functions
 - Triggers, indexes, procedures
-

Step 2: Practice Queries – Organized by Category










CAR Table Queries

Level: Basic to Advanced

1.  Find cars whose price is higher than the average price
 2.  Find the car with the second highest price
 3.  Find the third largest price (number)
 4.  Find the model of the car most available in count
 5.  Colors and count of cars
 6.  Cars ending with the letter 'H'
-

EMPLOYEE & DEPARTMENT Table Queries

Level: Intermediate to Advanced

1.  Find the second highest salary
2.  Find employees with more than 6 months of experience
3.  Find employees joined within 2 years
4.  Find the department with no employees
5.  Find the department with the highest average salary
6.  Count employees in the Engineering department
7.  Count how many employees each manager has
8.  Find the count of employees with the same salary
9.  Find department name and average salary

10. ☒ Show first name and department name of all employees
 11. ☒ Name and salary of employees with the highest salary
 12. ☒ Remove employees whose names end with 'n' or 'j'
 13. ☒ Remove employees with less than average salary
 14. ☒ Find the manager with the highest number of employees
 15. ☒ Update salary of employees whose name starts with 'D'
 16. ☒ Delete employees with salary > 80,000
 17. ☒ Delete a row from a table
 18. ☒ Delete all employees (delete without condition)
 19. ☒ Update by even number (salary or ID-based)
 20. ☒ SQL command to join and get latest 3 employees with department
 21. ☒ Update query using JOIN
 22. ☒ Queries using joins + subqueries + string functions + window functions
 23. ☒ Find the longest full name
-

CUSTOMER, PRODUCT, ORDERS Table Queries

Level: Intermediate to Advanced

1. ☒ Find customers who spent more than the average total spending
 2. ☒ Find customers who haven't placed any orders
 3. ☒ Find products with a price higher than the average price
 4. ☒ Find products not purchased by any customer
 5. ☒ Find the number of products purchased by each customer
 6. ☒ Find total order amount of each customer
-

STUDENT Table Queries

Level: Beginner to Intermediate

1. ☒ Group students by age and count
2. ☒ Filter groups with more than 2 students
3. ☒ Count of each age

4. ☒ Students not in departments X and Y
 5. ☒ Students with second largest age
-

Duplicate & Data Cleaning Queries

Level: Advanced

1. ☒ Find duplicate rows from a table
 2. ☒ Remove all duplicate records from a table
-
-

Step 1: Continue Using Existing Tables

We'll use your existing tables:

- car
 - employee
 - department
 - customer
 - product
 - orders
 - student
-

Step 2: Add Advanced Topics to the Practice Plan

Views Practice

1. ☒ Create a view including carname and price
↳ CREATE VIEW car_price AS SELECT model, price FROM car;
 2. ☒ Create a view with employee names starting with 'D'
↳ CREATE VIEW d_employees AS SELECT * FROM employee WHERE first_name LIKE 'D%';
 3. ☒ Create a view named car_price and display it
↳ SELECT * FROM car_price;
-

Stored Procedures & Functions

1. ☒ Create a stored procedure to increase salary by a fixed percent

2. ☒ Stored procedure to delete employees by name pattern
 3. ☒ Stored procedure to archive old orders
 4. ☒ Stored procedure with parameter (e.g., raise salary for dept)
 5. ☒ Stored procedure that inserts a log entry when a car is added
 6. ☒ Write a function in PostgreSQL (e.g., calculate yearly salary from monthly)
-

Triggers Practice

1. ☒ Create a trigger to log when a new car is added
 2. ☒ Trigger to auto-update total order amount in another table
 3. ☒ Trigger to prevent deletion of high-salary employees
-

Index Practice


1. ☒ Create an index on employee.salary
↳ CREATE INDEX idx_salary ON employee(salary);
 2. ☒ Create an index on car.model
↳ CREATE INDEX idx_model ON car(model);
 3. ☒ Create a composite index on orders(customer_id, product_id)
 4. ☒ List indexes in PostgreSQL
↳ \di or SELECT * FROM pg_indexes WHERE tablename = 'employee';
-


Transaction Practice





1. ☒ Implement a transaction to update two employees' salaries
 - Rollback on failure
 2. ☒ Transaction for order placement: update stock + insert order
 3. ☒ Transfer salary from one employee to another safely using a transaction
-

Step 3: Full Categorized Plan

Category	Questions
----------	-----------

 Basic SELECT & JOINS	Employee, Customer, Car, Product, Student queries
--	---

 Views	Create views with filters or projections
---	--

Category	Questions
 Stored Procedures	Procedures for update, delete, insert, logic
 Triggers	Create log triggers, prevent delete, automatic updates
 Indexes	Create & list indexes, performance optimization
 Transactions	Safe multi-step operations with rollback capability
