Table of Contents

1. Connecting to PostgreSQL

Start PostgreSQL Service

```
sudo systemctl start postgresql
```

Access PostgreSQL Shell

```
psql -U postgres
```

- -U – Specifies the user (default is postgres)
- Exit psql:

```
\q
```

Connect to a Database

```
\c database_name
```

---

2. Database Management

List All Databases

```
\l
```

Create a Database

```
CREATE DATABASE my_database;
```

Delete a Database

```
DROP DATABASE my_database;
```

Rename a Database

```
ALTER DATABASE old_name RENAME TO new_name;
```

---

3. User Management

List All Users

```
\du
```

Create a User

```
CREATE USER my_user WITH PASSWORD 'password123';
```

Grant All Privileges to a User

```
GRANT ALL PRIVILEGES ON DATABASE my_database TO my_user;
```

Delete a User

```
DROP USER my_user;
```

---

4. Table Management

List Tables

```
\dt
```

Create a Table

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    position VARCHAR(50),
    salary DECIMAL(10, 2),
    hire_date DATE
);
```

Delete a Table

```
DROP TABLE employees;
```

Add a Column

```
ALTER TABLE employees ADD email VARCHAR(100);
```

Delete a Column

```
ALTER TABLE employees DROP COLUMN email;
```

Rename a Table

```
ALTER TABLE employees RENAME TO staff;
```

Truncate (Empty) a Table

```
TRUNCATE TABLE employees;
```

---

5. Inserting Data

Insert a Single Row

```
INSERT INTO employees (name, position, salary, hire_date)
VALUES ('Alice', 'Manager', 75000, '2023-01-15');
```

Insert Multiple Rows

```
INSERT INTO employees (name, position, salary, hire_date) VALUES
('Bob', 'Developer', 60000, '2023-02-20'),
('Carol', 'Analyst', 58000, '2023-03-05');
```

---

6. Querying Data

Select All Data

```
SELECT * FROM employees;
```

Select Specific Columns

```
SELECT name, salary FROM employees;
```

Filter Data with WHERE Clause

```
SELECT * FROM employees WHERE salary > 60000;
```

Pattern Matching (LIKE)

```
SELECT * FROM employees WHERE name LIKE 'A%';
```

Sort Results (ORDER BY)

```
SELECT * FROM employees ORDER BY salary DESC;
```

Limit Results

```
SELECT * FROM employees LIMIT 5;
```

Pagination (LIMIT + OFFSET)

```
SELECT * FROM employees LIMIT 5 OFFSET 10;
```

Distinct Values

```
SELECT DISTINCT position FROM employees;
```

---

7. Updating Data

Update Specific Rows

```
UPDATE employees
SET salary = 80000
WHERE name = 'Alice';
```

Update Multiple Rows

```
UPDATE employees
SET position = 'Senior Developer'
```

```
WHERE position = 'Developer';
```

---

8. Deleting Data

Delete Specific Rows

```
DELETE FROM employees WHERE name = 'Bob';
```

Delete All Rows

```
DELETE FROM employees;
```

---

9. Aggregation and Grouping

Aggregate Functions

```
SELECT COUNT(*) FROM employees;         -- Count
SELECT AVG(salary) FROM employees;      -- Average
SELECT MAX(salary) FROM employees;      -- Maximum
SELECT MIN(salary) FROM employees;      -- Minimum
SELECT SUM(salary) FROM employees;      -- Sum
```

Group Data

```
SELECT position, COUNT(*) FROM employees GROUP BY position;
```

Filter Grouped Data (HAVING)

```
SELECT position, AVG(salary)
```

```
FROM employees
GROUP BY position
HAVING AVG(salary) > 60000;
```

---

10. Table Joins

Inner Join

```
SELECT employees.name, departments.department_name
FROM employees
INNER JOIN departments
ON employees.department_id = departments.id;
```

Left Join

```
SELECT employees.name, departments.department_name
FROM employees
LEFT JOIN departments
ON employees.department_id = departments.id;
```

Right Join

```
SELECT employees.name, departments.department_name
FROM employees
RIGHT JOIN departments
ON employees.department_id = departments.id;
```

Full Outer Join

```
SELECT employees.name, departments.department_name
FROM employees
```

```
FULL OUTER JOIN departments
ON employees.department_id = departments.id;
```

---

11. Subqueries

Subquery in WHERE Clause

```
SELECT name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

Subquery in SELECT

```
SELECT name,
       (SELECT AVG(salary) FROM employees) AS average_salary
FROM employees;
```

---

12. Indexing

Create an Index

```
CREATE INDEX idx_salary ON employees(salary);
```

Drop an Index

```
DROP INDEX idx_salary;
```

---

13. Backup and Restore

Backup a Database

```
pg_dump my_database > backup.sql
```

Restore a Database

```
psql my_database < backup.sql
```

---

14. Useful Meta-Commands

Command	Description
\l	List all databases
\c dbname	Connect to a database
\dt	List tables in the current database
\d table_name	Describe a table
\du	List all users
\q	Exit psql
\df	List all functions
\x	Toggle extended display mode
\conninfo	Display current connection info

Tips for PostgreSQL

- Use transactions to ensure data consistency (BEGIN and COMMIT).
- Backup frequently to avoid data loss.
- Indexes improve query performance for large datasets.
- Always test queries in a safe environment before applying to production.