PostgreSQL Complete Cheat Sheet

Date & Time Operations

- CURRENT_DATE, CURRENT_TIME, NOW(), CURRENT_TIMESTAMP
- Date arithmetic: + INTERVAL '1 day', - INTERVAL '1 week'
- EXTRACT(YEAR FROM ...), DATE_TRUNC('month', NOW())
- AGE(), date1 - date2 returns days

String Functions & Slicing

- SUBSTRING('Postgres' FROM 1 FOR 4)
- LEFT/RIGHT: LEFT('Hello', 2)
- POSITION, REPLACE, UPPER, LOWER, TRIM, LTRIM, RTRIM
- LENGTH('abc'), || for concatenation

Math Functions

- ABS(), ROUND(), CEIL(), FLOOR(), POWER(), SQRT(), MOD()

Pattern Matching

- LIKE, ILIKE for case-insensitive search
- SIMILAR TO, ~ for regex
- IN, BETWEEN, NOT IN

Aggregate Functions

- COUNT(), SUM(), AVG(), MIN(), MAX()

Joins & Subqueries

- INNER JOIN, LEFT JOIN, RIGHT JOIN
- Subquery in WHERE: SELECT ... WHERE id IN (SELECT ...)

Creating Tables & Constraints

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT UNIQUE,
    salary NUMERIC(10,2),
    hire_date DATE DEFAULT CURRENT_DATE
```

```
);
```

Constraints Explained

```
- NOT NULL, UNIQUE, DEFAULT, CHECK, PRIMARY KEY, FOREIGN KEY
- Composite keys: PRIMARY KEY(col1, col2)
- Foreign key with ON DELETE CASCADE
```

Altering Tables

```
- ALTER TABLE ADD/DROP/RENAME COLUMN
- ALTER TABLE RENAME TO new_name
```

ENUMs and Domains

```
CREATE TYPE gender AS ENUM('male','female','other');
CREATE DOMAIN positive_int AS INT CHECK (VALUE > 0);
```

Casting & Conversion

```
- '123'::INTEGER, 100::TEXT, CAST(value AS type)
```

Window Functions & Extras

```
- RANK() OVER (PARTITION BY ... ORDER BY ...)
- COALESCE(), NULLIF()
```

Referencing Primary Keys (Foreign Keys)

```
CREATE TABLE departments (
    dept_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL
);

CREATE TABLE employees (
    emp_id SERIAL PRIMARY KEY,
    emp_name TEXT NOT NULL,
    dept_id INT REFERENCES departments(dept_id) ON DELETE CASCADE
);
```

LIKE, ILIKE, and Pattern Matching

PostgreSQL Complete Cheat Sheet

```
- LIKE: case-sensitive pattern match
  SELECT * FROM users WHERE name LIKE 'J%';

- ILIKE: case-insensitive match
  SELECT * FROM users WHERE name ILIKE '%doe%';

- SIMILAR TO: uses SQL regex-like patterns
  SELECT * FROM text_table WHERE text SIMILAR TO '(abc|def)%';
```

GROUP BY and HAVING

```
- GROUP BY: groups rows with same values
  SELECT dept_id, COUNT(*) FROM employees GROUP BY dept_id;

- HAVING: filters groups
  SELECT dept_id, COUNT(*) FROM employees GROUP BY dept_id HAVING COUNT(*) > 5;
```

Subqueries

```
- In SELECT:
  SELECT name, (SELECT AVG(salary) FROM employees) AS avg_salary FROM employees;

- In WHERE:
   SELECT name FROM employees WHERE dept_id IN (SELECT dept_id FROM departments WHERE
name = 'HR');

- Correlated:
   SELECT name FROM employees e WHERE salary > (SELECT AVG(salary) FROM employees WHERE
dept_id = e.dept_id);
```

Views

```
- Create a virtual table (stored query):
  CREATE VIEW hr_employees AS
   SELECT name, salary FROM employees WHERE dept_id = (SELECT dept_id FROM departments
WHERE name='HR');

- Query a view:
  SELECT * FROM hr_employees;

- Drop view:
  DROP VIEW hr_employees;
```

User-defined Functions

PostgreSQL Complete Cheat Sheet

```
CREATE OR REPLACE FUNCTION get_bonus(salary NUMERIC)
RETURNS NUMERIC AS $$
BEGIN
  RETURN salary * 0.10;
END;
$$ LANGUAGE plpgsql;

-- Usage:
SELECT name, get_bonus(salary) FROM employees;
```