CREATE TABLE IF NOT EXISTS phonebook (

id SERIAL PRIMARY KEY,

first\_name VARCHAR(100),

phone\_number VARCHAR(20)

);

CREATE OR REPLACE FUNCTION search\_pattern(pattern TEXT)

RETURNS TABLE (id INT, first\_name VARCHAR, phone\_number VARCHAR)

AS $$

BEGIN

RETURN QUERY

SELECT p.id, p.first\_name, p.phone\_number

FROM phonebook p

WHERE p.first\_name ILIKE '%' || pattern || '%'

OR p.phone\_number LIKE '%' || pattern || '%';

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE PROCEDURE insert\_or\_update\_user(name TEXT, phone TEXT)

AS $$

BEGIN

IF EXISTS (SELECT 1 FROM phonebook WHERE first\_name = name) THEN

UPDATE phonebook SET phone\_number = phone WHERE first\_name = name;

ELSE

INSERT INTO phonebook (first\_name, phone\_number) VALUES (name, phone);

END IF;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE PROCEDURE insert\_many\_users(names TEXT[], phones TEXT[])

AS $$

DECLARE

i INTEGER;

invalid TEXT := '';

BEGIN

FOR i IN 1..array\_length(names, 1) LOOP

IF phones[i] ~ '^\d{10,15}$' THEN

PERFORM insert\_or\_update\_user(names[i], phones[i]);

ELSE

RAISE NOTICE 'Invalid phone for %: %', names[i], phones[i];

END IF;

END LOOP;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_users\_paginated(\_limit INT, \_offset INT)

RETURNS TABLE (id INT, first\_name VARCHAR, phone\_number VARCHAR)

AS $$

BEGIN

RETURN QUERY

SELECT p.id, p.first\_name, p.phone\_number

FROM phonebook p

ORDER BY p.id

LIMIT \_limit OFFSET \_offset;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE PROCEDURE delete\_by\_name\_or\_phone(name TEXT, phone TEXT)

AS $$

BEGIN

DELETE FROM phonebook

WHERE (name IS NOT NULL AND first\_name = name)

OR (phone IS NOT NULL AND phone\_number = phone);

END;

$$ LANGUAGE plpgsql;