Advanced SQL

This document contains an extensive, high-performance, and real-world SQL sample designed to demonstrate mastery of complex SQL features, best practices in data modeling, query optimization, and real-world business logic.

SQL Sample Code

```
-- SCHEMA DESIGN: E-Commerce System

-- DROP TABLES IF EXISTS
DROP TABLE IF EXISTS order_items, orders, payments, customers, products, categories, product_reviews CASCADE;

-- 1. Customers Table
CREATE TABLE customers (
    customer_id SERIAL PRIMARY KEY,
    full_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 2. Categories Table (for recursive hierarchy)
CREATE TABLE categories (
    category_id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    parent_category_id INT REFERENCES categories(category_id) ON DELETE SET NULL
);

-- 3. Products Table
CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price NUMERIC(10,2) NOT NULL CHECK (price >= 0),
    stock_quantity INT NOT NULL DEFAULT 0 CHECK (stock_quantity >= 0),
    category_id INT REFERENCES categories(category_id) ON DELETE SET NULL,
```

```
    specifications JSONB
);

-- 4. Orders Table
CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    customer_id INT REFERENCES customers(customer_id),
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) DEFAULT 'PENDING' CHECK (status IN ('PENDING',
'SHIPPED', 'DELIVERED', 'CANCELLED')),
    total_amount NUMERIC(12,2)
);

-- 5. Order Items Table
CREATE TABLE order_items (
    order_item_id SERIAL PRIMARY KEY,
    order_id INT REFERENCES orders(order_id) ON DELETE CASCADE,
    product_id INT REFERENCES products(product_id),
    quantity INT NOT NULL CHECK (quantity > 0),
    unit_price NUMERIC(10,2) NOT NULL
);

-- 6. Payments Table
CREATE TABLE payments (
    payment_id SERIAL PRIMARY KEY,
    order_id INT REFERENCES orders(order_id),
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    amount NUMERIC(12,2) NOT NULL,
    payment_method VARCHAR(20) CHECK (payment_method IN ('CREDIT_CARD',
'PAYPAL', 'BANK_TRANSFER')),
    status VARCHAR(20) DEFAULT 'COMPLETED' CHECK (status IN ('PENDING',
'COMPLETED', 'FAILED'))
);

-- 7. Product Reviews Table
CREATE TABLE product_reviews (
    review_id SERIAL PRIMARY KEY,
    customer_id INT REFERENCES customers(customer_id),
    product_id INT REFERENCES products(product_id),
    rating INT CHECK (rating BETWEEN 1 AND 5),
```

```
    review TEXT,
    review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Example Queries and Features

-- Find top 5 most sold products
SELECT p.name, SUM(oi.quantity) AS total_sold
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.name
ORDER BY total_sold DESC
LIMIT 5;

-- Recursive CTE: Category Hierarchy
WITH RECURSIVE category_tree AS (
    SELECT category_id, name, parent_category_id, 1 AS level
    FROM categories
    WHERE parent_category_id IS NULL
    UNION ALL
    SELECT c.category_id, c.name, c.parent_category_id, ct.level + 1
    FROM categories c
    JOIN category_tree ct ON c.parent_category_id = ct.category_id
)
SELECT * FROM category_tree;

-- Stored Procedure Example
CREATE OR REPLACE FUNCTION place_order(customer_id INT, items JSONB)
RETURNS VOID AS $$
DECLARE
    order_id INT;
BEGIN
    INSERT INTO orders (customer_id, status, total_amount)
    VALUES (customer_id, 'PENDING', 0)
    RETURNING order_id INTO order_id;

    INSERT INTO order_items (order_id, product_id, quantity, unit_price)
    SELECT order_id, (item->>'product_id')::INT, (item->>'quantity')::INT,
        (SELECT price FROM products WHERE product_id = (item-
>>'product_id')::INT)
```

```
    FROM jsonb_array_elements(items) AS item;

    UPDATE orders
    SET total_amount = (SELECT SUM(quantity * unit_price) FROM order_items
WHERE order_id = orders.order_id)
    WHERE order_id = order_id;
END;
$$ LANGUAGE plpgsql;
```