```
-- =====================================
-- SCRIPT DE CONFIGURACIÓN INICIAL
-- Base de Datos: Aplicación de Gestión de Deudas
-- =====================================

-- 1. Crear la base de datos (ejecutar como superusuario)
-- CREATE DATABASE debt_management_app;
-- \c debt_management_app;

-- 2. Crear extensiones útiles
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- =====================================
-- CREACIÓN DE TABLAS
-- =====================================

-- Tabla de usuarios
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  email VARCHAR(255) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Tabla de deudas
CREATE TABLE debts (
  id SERIAL PRIMARY KEY,
  user_id INTEGER NOT NULL,
  debtor_id INTEGER NULL,
  title VARCHAR(200) NOT NULL,
  description TEXT,
  amount DECIMAL(12,2) NOT NULL CHECK (amount > 0),
  currency VARCHAR(3) DEFAULT 'COP',
  is_paid BOOLEAN DEFAULT FALSE,
  due_date TIMESTAMP WITH TIME ZONE,
  paid_at TIMESTAMP WITH TIME ZONE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

  -- Foreign Keys
  CONSTRAINT fk_debts_user FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE
CASCADE,
  CONSTRAINT fk_debts_debtor FOREIGN KEY (debtor_id) REFERENCES users(id) ON DELETE
SET NULL,

  -- Restricciones de negocio
  CONSTRAINT check_paid_date CHECK (
      (is_paid = FALSE AND paid_at IS NULL) OR
      (is_paid = TRUE AND paid_at IS NOT NULL)
  )
);

-- ====================================
-- ÍNDICES PARA OPTIMIZACIÓN
-- ====================================
-- Índices en users
CREATE UNIQUE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_created_at ON users(created_at);

-- Índices en debts
CREATE INDEX idx_debts_user_id ON debts(user_id);
CREATE INDEX idx_debts_debtor_id ON debts(debtor_id);
CREATE INDEX idx_debts_is_paid ON debts(is_paid);
CREATE INDEX idx_debts_user_paid ON debts(user_id, is_paid);
CREATE INDEX idx_debts_created_at ON debts(created_at);
CREATE INDEX idx_debts_due_date ON debts(due_date) WHERE due_date IS NOT NULL;

-- ====================================
-- FUNCIÓN PARA AUTO-UPDATE timestamp
-- ====================================

-- Función para actualizar el campo updated_at automáticamente
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
   NEW.updated_at = CURRENT_TIMESTAMP;
   RETURN NEW;
END;
$$ language 'plpgsql';

-- Triggers para actualización automática
CREATE TRIGGER update_users_updated_at
   BEFORE UPDATE ON users
   FOR EACH ROW
```

```sql
    EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_debts_updated_at
    BEFORE UPDATE ON debts
    FOR EACH ROW
    EXECUTE FUNCTION update_updated_at_column();

-- ====================================
-- DATOS DE PRUEBA (OPCIONAL)
-- ====================================

-- Usuario de prueba
INSERT INTO users (email, password_hash, first_name, last_name) VALUES
('test@example.com', '$2a$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', 'Juan', 'Pérez'),
('maria@example.com', '$2a$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', 'María', 'González');

-- Deudas de prueba
INSERT INTO debts (user_id, title, description, amount, currency) VALUES
(1, 'Préstamo a Carlos', 'Dinero prestado para emergencia médica', 150000.00, 'COP'),
(1, 'Cena en restaurante', 'Dividir cuenta de cena grupal', 45000.00, 'COP'),
(2, 'Préstamo para auto', 'Ayuda para enganche del vehículo', 2000000.00, 'COP');

-- ====================================
-- CONSULTAS DE VERIFICACIÓN
-- ====================================

-- Verificar que todo se creó correctamente
SELECT 'Users count: ' || COUNT(*) as result FROM users
```

```
UNION ALL

SELECT 'Debts count: ' || COUNT(*) as result FROM debts

UNION ALL

SELECT 'Indexes created: ' || COUNT(*) as result FROM pg_indexes WHERE tablename IN
('users', 'debts');


-- Consulta de prueba: deudas pendientes con información del usuario

SELECT

  u.first_name || ' ' || u.last_name as owner,

  d.title,

  d.amount,

  d.currency,

  d.created_at

FROM debts d

JOIN users u ON d.user_id = u.id

WHERE d.is_paid = FALSE

ORDER BY d.created_at DESC;
```