# Scripts SQL para Configuração do Supabase

## Instruções de Execução

1. Acesse o dashboard do seu projeto Supabase
2. Vá para "SQL Editor"
3. Execute primeiro o script `schema.sql`
4. Execute depois o script `sample_data.sql` (opcional, para dados de demonstração)

## Schema Principal (schema.sql)

Execute este script primeiro para criar toda a estrutura do banco:

```
 -- ========================================
-- SISTEMA DE GESTÃO DE PESSOAS - SCHEMA
-- ========================================

-- Extensões necessárias
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "pgcrypto";

-- Função para atualizar timestamp
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ language 'plpgsql';

-- ========================================
-- TABELAS PRINCIPAIS
-- ========================================

-- Tabela de usuários
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    name VARCHAR(255) NOT NULL,
    role VARCHAR(50) NOT NULL DEFAULT 'employee' CHECK (role IN ('admin',
```

```
'manager', 'employee')),
    is_active BOOLEAN DEFAULT true,
    last_login TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de departamentos
CREATE TABLE departments (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    manager_id UUID REFERENCES users(id),
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de cargos
CREATE TABLE positions (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    salary DECIMAL(10,2),
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de perfis de usuário
CREATE TABLE user_profiles (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID UNIQUE REFERENCES users(id) ON DELETE CASCADE,
    phone VARCHAR(20),
    address TEXT,
    birth_date DATE,
    hire_date DATE,
    position_id UUID REFERENCES positions(id),
    department_id UUID REFERENCES departments(id),
    emergency_contact_name VARCHAR(255),
    emergency_contact_phone VARCHAR(20),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de avaliações de desempenho
CREATE TABLE performance_evaluations (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    evaluator_id UUID REFERENCES users(id),
    period_start DATE NOT NULL,
    period_end DATE NOT NULL,
```

```
    overall_score DECIMAL(3,2) CHECK (overall_score >= 0 AND overall_score <= 10),
    goals_achievement DECIMAL(3,2) CHECK (goals_achievement >= 0 AND
goals_achievement <= 10),
    technical_skills DECIMAL(3,2) CHECK (technical_skills >= 0 AND technical_skills
<= 10),
    soft_skills DECIMAL(3,2) CHECK (soft_skills >= 0 AND soft_skills <= 10),
    comments TEXT,
    feedback TEXT,
    status VARCHAR(20) DEFAULT 'draft' CHECK (status IN ('draft', 'submitted',
'approved', 'rejected')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de treinamentos
CREATE TABLE trainings (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    title VARCHAR(255) NOT NULL,
    description TEXT,
    instructor VARCHAR(255),
    duration_hours INTEGER,
    max_participants INTEGER,
    start_date TIMESTAMP,
    end_date TIMESTAMP,
    location VARCHAR(255),
    type VARCHAR(50) DEFAULT 'internal' CHECK (type IN ('internal', 'external',
'online')),
    status VARCHAR(20) DEFAULT 'scheduled' CHECK (status IN ('scheduled',
'in_progress', 'completed', 'cancelled')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de participações em treinamentos
CREATE TABLE training_participations (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    training_id UUID REFERENCES trainings(id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    status VARCHAR(20) DEFAULT 'enrolled' CHECK (status IN ('enrolled',
'completed', 'cancelled')),
    completion_date DATE,
    score DECIMAL(3,2),
    certificate_url VARCHAR(500),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(training_id, user_id)
);

-- Tabela de férias
CREATE TABLE vacations (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
```

```
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    days_requested INTEGER NOT NULL,
    type VARCHAR(20) DEFAULT 'annual' CHECK (type IN ('annual', 'sick', 'personal',
'maternity', 'paternity')),
    status VARCHAR(20) DEFAULT 'pending' CHECK (status IN ('pending', 'approved',
'rejected', 'cancelled')),
    approved_by UUID REFERENCES users(id),
    approved_at TIMESTAMP,
    comments TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de benefícios
CREATE TABLE benefits (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    type VARCHAR(50) DEFAULT 'other' CHECK (type IN ('health', 'dental',
'transport', 'meal', 'education', 'other')),
    value DECIMAL(10,2),
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de benefícios dos usuários
CREATE TABLE user_benefits (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    benefit_id UUID REFERENCES benefits(id) ON DELETE CASCADE,
    start_date DATE NOT NULL,
    end_date DATE,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(user_id, benefit_id)
);

-- Tabela de admissões
CREATE TABLE admissions (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    status VARCHAR(20) DEFAULT 'pending' CHECK (status IN ('pending',
'in_progress', 'completed', 'cancelled')),
    documents_received BOOLEAN DEFAULT false,
    background_check BOOLEAN DEFAULT false,
    equipment_assigned BOOLEAN DEFAULT false,
    system_access_granted BOOLEAN DEFAULT false,
    orientation_completed BOOLEAN DEFAULT false,
    assigned_to UUID REFERENCES users(id),
```

```
    notes TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de mensagens do chat
CREATE TABLE chat_messages (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    message TEXT NOT NULL,
    message_type VARCHAR(20) DEFAULT 'user' CHECK (message_type IN ('user',
'bot')),
    sender_id UUID REFERENCES users(id),
    recipient_id UUID REFERENCES users(id),
    is_read BOOLEAN DEFAULT false,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela de configurações do sistema
CREATE TABLE system_settings (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    key VARCHAR(255) UNIQUE NOT NULL,
    value TEXT,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- ========================================
-- ÍNDICES PARA PERFORMANCE
-- ========================================

-- Índices para users
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_role ON users(role);
CREATE INDEX idx_users_active ON users(is_active);

-- Índices para user_profiles
CREATE INDEX idx_user_profiles_user_id ON user_profiles(user_id);
CREATE INDEX idx_user_profiles_department ON user_profiles(department_id);
CREATE INDEX idx_user_profiles_position ON user_profiles(position_id);

-- Índices para performance_evaluations
CREATE INDEX idx_performance_user_id ON performance_evaluations(user_id);
CREATE INDEX idx_performance_evaluator ON
performance_evaluations(evaluator_id);
CREATE INDEX idx_performance_period ON
performance_evaluations(period_start, period_end);
CREATE INDEX idx_performance_status ON performance_evaluations(status);

-- Índices para trainings
CREATE INDEX idx_trainings_status ON trainings(status);
```

```sql
CREATE INDEX idx_trainings_dates ON trainings(start_date, end_date);

-- Índices para training_participations
CREATE INDEX idx_training_participations_training ON
training_participations(training_id);
CREATE INDEX idx_training_participations_user ON
training_participations(user_id);

-- Índices para vacations
CREATE INDEX idx_vacations_user_id ON vacations(user_id);
CREATE INDEX idx_vacations_status ON vacations(status);
CREATE INDEX idx_vacations_dates ON vacations(start_date, end_date);

-- Índices para chat_messages
CREATE INDEX idx_chat_sender ON chat_messages(sender_id);
CREATE INDEX idx_chat_recipient ON chat_messages(recipient_id);
CREATE INDEX idx_chat_created_at ON chat_messages(created_at);
CREATE INDEX idx_chat_unread ON chat_messages(recipient_id, is_read) WHERE
is_read = false;

-- ========================================
-- TRIGGERS PARA UPDATED_AT
-- ========================================

CREATE TRIGGER update_users_updated_at BEFORE UPDATE ON users
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_departments_updated_at BEFORE UPDATE ON
departments
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_positions_updated_at BEFORE UPDATE ON positions
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_user_profiles_updated_at BEFORE UPDATE ON
user_profiles
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_performance_evaluations_updated_at BEFORE UPDATE
ON performance_evaluations
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_trainings_updated_at BEFORE UPDATE ON trainings
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_training_participations_updated_at BEFORE UPDATE ON
training_participations
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_vacations_updated_at BEFORE UPDATE ON vacations
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();
```

```sql
CREATE TRIGGER update_benefits_updated_at BEFORE UPDATE ON benefits
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_user_benefits_updated_at BEFORE UPDATE ON
user_benefits
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_admissions_updated_at BEFORE UPDATE ON admissions
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_chat_messages_updated_at BEFORE UPDATE ON
chat_messages
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_system_settings_updated_at BEFORE UPDATE ON
system_settings
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

-- ======================================
-- VIEWS PARA DASHBOARD E RELATÓRIOS
-- ======================================

-- View para estatísticas do dashboard
CREATE VIEW dashboard_stats AS
SELECT
    (SELECT COUNT(*) FROM users WHERE is_active = true) as total_employees,
    (SELECT COUNT(*) FROM departments WHERE is_active = true) as
total_departments,
    (SELECT COUNT(*) FROM positions WHERE is_active = true) as total_positions,
    (SELECT COUNT(*) FROM vacations WHERE status = 'pending') as
pending_vacations,
    (SELECT COUNT(*) FROM performance_evaluations WHERE status = 'pending')
as pending_evaluations,
    (SELECT COUNT(*) FROM trainings WHERE status = 'scheduled' AND start_date
> CURRENT_DATE) as upcoming_trainings;

-- View para funcionários com detalhes
CREATE VIEW employees_detailed AS
SELECT
    u.id,
    u.email,
    u.name,
    u.role,
    u.is_active,
    u.last_login,
    up.phone,
    up.hire_date,
    p.name as position_name,
    p.salary,
    d.name as department_name,
    d.manager_id as department_manager_id,
    u.created_at,
```

```sql
    u.updated_at
FROM users u
LEFT JOIN user_profiles up ON u.id = up.user_id
LEFT JOIN positions p ON up.position_id = p.id
LEFT JOIN departments d ON up.department_id = d.id;

-- View para avaliações com detalhes
CREATE VIEW evaluations_detailed AS
SELECT
    pe.id,
    pe.period_start,
    pe.period_end,
    pe.overall_score,
    pe.goals_achievement,
    pe.technical_skills,
    pe.soft_skills,
    pe.status,
    pe.comments,
    pe.feedback,
    u.name as employee_name,
    u.email as employee_email,
    ev.name as evaluator_name,
    ev.email as evaluator_email,
    d.name as department_name,
    p.name as position_name,
    pe.created_at,
    pe.updated_at
FROM performance_evaluations pe
JOIN users u ON pe.user_id = u.id
LEFT JOIN users ev ON pe.evaluator_id = ev.id
LEFT JOIN user_profiles up ON u.id = up.user_id
LEFT JOIN departments d ON up.department_id = d.id
LEFT JOIN positions p ON up.position_id = p.id;

-- View para treinamentos com participações
CREATE VIEW trainings_with_participants AS
SELECT
    t.id,
    t.title,
    t.description,
    t.instructor,
    t.duration_hours,
    t.max_participants,
    t.start_date,
    t.end_date,
    t.location,
    t.type,
    t.status,
    COUNT(tp.id) as enrolled_count,
    COUNT(CASE WHEN tp.status = 'completed' THEN 1 END) as completed_count,
    t.created_at,
    t.updated_at
```

```sql
FROM trainings t
LEFT JOIN training_participations tp ON t.id = tp.training_id
GROUP BY t.id, t.title, t.description, t.instructor, t.duration_hours,
         t.max_participants, t.start_date, t.end_date, t.location,
         t.type, t.status, t.created_at, t.updated_at;

-- ========================================
-- FUNÇÕES UTILITÁRIAS
-- ========================================

-- Função para calcular dias de férias disponíveis
CREATE OR REPLACE FUNCTION calculate_vacation_days(user_id UUID, year
INTEGER DEFAULT EXTRACT(YEAR FROM CURRENT_DATE))
RETURNS INTEGER AS $$
DECLARE
    hire_date DATE;
    days_used INTEGER;
    days_available INTEGER;
BEGIN
    -- Buscar data de contratação
    SELECT up.hire_date INTO hire_date
    FROM user_profiles up
    WHERE up.user_id = calculate_vacation_days.user_id;

    IF hire_date IS NULL THEN
        RETURN 0;
    END IF;

    -- Calcular dias disponíveis (30 dias por ano, proporcional se contratado no ano)
    IF EXTRACT(YEAR FROM hire_date) = year THEN
        days_available := FLOOR((12 - EXTRACT(MONTH FROM hire_date) + 1) * 2.5);
    ELSE
        days_available := 30;
    END IF;

    -- Calcular dias já utilizados no ano
    SELECT COALESCE(SUM(days_requested), 0) INTO days_used
    FROM vacations v
    WHERE v.user_id = calculate_vacation_days.user_id
    AND v.status = 'approved'
    AND EXTRACT(YEAR FROM v.start_date) = year;

    RETURN GREATEST(0, days_available - days_used);
END;
$$ LANGUAGE plpgsql;

-- Função para obter próximas avaliações
CREATE OR REPLACE FUNCTION get_upcoming_evaluations(days_ahead INTEGER
DEFAULT 30)
RETURNS TABLE(
    user_id UUID,
    user_name VARCHAR,
```

```
    last_evaluation_date DATE,
    days_since_last_evaluation INTEGER
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        u.id,
        u.name,
        MAX(pe.period_end) as last_evaluation_date,
        (CURRENT_DATE - MAX(pe.period_end))::INTEGER as
days_since_last_evaluation
    FROM users u
    LEFT JOIN performance_evaluations pe ON u.id = pe.user_id
    WHERE u.is_active = true
    GROUP BY u.id, u.name
    HAVING MAX(pe.period_end) IS NULL
        OR (CURRENT_DATE - MAX(pe.period_end)) >= days_ahead;
END;
$$ LANGUAGE plpgsql;

-- ========================================
-- CONFIGURAÇÕES INICIAIS
-- ========================================

-- Inserir configurações padrão do sistema
INSERT INTO system_settings (key, value, description) VALUES
('company_name', 'Empresa LTDA', 'Nome da empresa'),
('vacation_days_per_year', '30', 'Dias de férias por ano'),
('evaluation_frequency_months', '12', 'Frequência de avaliações em meses'),
('chat_bot_enabled', 'true', 'Habilitar chat bot'),
('max_file_upload_size', '10485760', 'Tamanho máximo de upload em bytes
(10MB)');

-- Criar usuário administrador padrão (senha: admin123)
INSERT INTO users (email, password_hash, name, role) VALUES
('admin@hrmanagement.com',
'$2b$10$rQZ8kHWiZ8qHZqHZqHZqHOeKqHZqHZqHZqHZqHZqHZqHZqHZqH',
'Administrador', 'admin');

COMMIT;
```

## Dados de Exemplo (sample_data.sql)

Execute este script após o schema para inserir dados de demonstração:

```
 -- ========================================
-- DADOS DE EXEMPLO PARA DEMONSTRAÇÃO
-- ========================================
```

```sql
-- Departamentos
INSERT INTO departments (id, name, description) VALUES
(uuid_generate_v4(), 'Tecnologia da Informação', 'Departamento responsável por desenvolvimento e infraestrutura'),
(uuid_generate_v4(), 'Recursos Humanos', 'Departamento de gestão de pessoas'),
(uuid_generate_v4(), 'Financeiro', 'Departamento financeiro e contábil'),
(uuid_generate_v4(), 'Marketing', 'Departamento de marketing e vendas'),
(uuid_generate_v4(), 'Operações', 'Departamento de operações e logística');

-- Cargos
INSERT INTO positions (id, name, description, salary) VALUES
(uuid_generate_v4(), 'Desenvolvedor Full Stack',
'Desenvolvedor com conhecimento em frontend e backend', 8000.00),
(uuid_generate_v4(), 'Analista de RH', 'Analista responsável por processos de RH',
5500.00),
(uuid_generate_v4(), 'Gerente de TI', 'Gerente do departamento de TI', 12000.00),
(uuid_generate_v4(), 'Analista Financeiro', 'Analista responsável por análises financeiras', 6000.00),
(uuid_generate_v4(), 'Coordenador de Marketing', 'Coordenador de campanhas de marketing', 7000.00),
(uuid_generate_v4(), 'Assistente Administrativo', 'Assistente para atividades administrativas', 3500.00),
(uuid_generate_v4(), 'Designer UX/UI', 'Designer de interfaces e experiência do usuário', 7500.00),
(uuid_generate_v4(), 'Analista de Dados', 'Analista responsável por análise de dados', 8500.00);

-- Usuários de exemplo
DO $$
DECLARE
    dept_ti_id UUID;
    dept_rh_id UUID;
    dept_fin_id UUID;
    dept_mkt_id UUID;
    pos_dev_id UUID;
    pos_analista_rh_id UUID;
    pos_gerente_ti_id UUID;
    pos_analista_fin_id UUID;
    pos_coord_mkt_id UUID;
    pos_designer_id UUID;
    user_admin_id UUID;
    user_manager_id UUID;
    user_emp1_id UUID;
    user_emp2_id UUID;
    user_emp3_id UUID;
BEGIN
    -- Buscar IDs dos departamentos
    SELECT id INTO dept_ti_id FROM departments WHERE name = 'Tecnologia da Informação';
    SELECT id INTO dept_rh_id FROM departments WHERE name = 'Recursos Humanos';
    SELECT id INTO dept_fin_id FROM departments WHERE name = 'Financeiro';
```

```
    SELECT id INTO dept_mkt_id FROM departments WHERE name = 'Marketing';

    -- Buscar IDs dos cargos
    SELECT id INTO pos_dev_id FROM positions WHERE name = 'Desenvolvedor Full
Stack';
    SELECT id INTO pos_analista_rh_id FROM positions WHERE name = 'Analista de
RH';
    SELECT id INTO pos_gerente_ti_id FROM positions WHERE name = 'Gerente de
TI';
    SELECT id INTO pos_analista_fin_id FROM positions WHERE name = 'Analista
Financeiro';
    SELECT id INTO pos_coord_mkt_id FROM positions WHERE name =
'Coordenador de Marketing';
    SELECT id INTO pos_designer_id FROM positions WHERE name = 'Designer UX/
UI';

    -- Buscar ID do admin
    SELECT id INTO user_admin_id FROM users WHERE email =
'admin@hrmanagement.com';

    -- Inserir usuários adicionais
    INSERT INTO users (id, email, password_hash, name, role) VALUES
    (uuid_generate_v4(), 'joao.silva@empresa.com',
'$2b$10$rQZ8kHWiZ8qHZqHZqHZqHOeKqHZqHZqHZqHZqHZqHZqHZqHZqHZqH',
'João Silva', 'manager'),
    (uuid_generate_v4(), 'maria.santos@empresa.com',
'$2b$10$rQZ8kHWiZ8qHZqHZqHZqHOeKqHZqHZqHZqHZqHZqHZqHZqHZqHZqH',
'Maria Santos', 'employee'),
    (uuid_generate_v4(), 'pedro.oliveira@empresa.com',
'$2b$10$rQZ8kHWiZ8qHZqHZqHZqHOeKqHZqHZqHZqHZqHZqHZqHZqHZqHZqH',
'Pedro Oliveira', 'employee'),
    (uuid_generate_v4(), 'ana.costa@empresa.com',
'$2b$10$rQZ8kHWiZ8qHZqHZqHZqHZqHZqHZqHZqHZqHZqHZqHZqHZqHZqHZqH',
'Ana Costa', 'employee'),
    (uuid_generate_v4(), 'carlos.ferreira@empresa.com',
'$2b$10$rQZ8kHWiZ8qHZqHZqHZqHOeKqHZqHZqHZqHZqHZqHZqHZqHZqHZqH',
'Carlos Ferreira', 'employee');

    -- Buscar IDs dos novos usuários
    SELECT id INTO user_manager_id FROM users WHERE email =
'joao.silva@empresa.com';
    SELECT id INTO user_emp1_id FROM users WHERE email =
'maria.santos@empresa.com';
    SELECT id INTO user_emp2_id FROM users WHERE email =
'pedro.oliveira@empresa.com';
    SELECT id INTO user_emp3_id FROM users WHERE email =
'ana.costa@empresa.com';

    -- Atualizar manager do departamento de TI
    UPDATE departments SET manager_id = user_manager_id WHERE id = dept_ti_id;

    -- Inserir perfis dos usuários
```

```
    INSERT INTO user_profiles (user_id, phone, address, birth_date, hire_date,
position_id, department_id, emergency_contact_name, emergency_contact_phone)
VALUES
    (user_admin_id, '(11) 99999-0000', 'São Paulo, SP', '1985-01-15', '2020-01-01',
pos_gerente_ti_id, dept_ti_id, 'Contato Admin', '(11) 88888-0000'),
    (user_manager_id, '(11) 99999-1111', 'São Paulo, SP', '1980-05-20', '2021-03-15',
pos_gerente_ti_id, dept_ti_id, 'Esposa João', '(11) 88888-1111'),
    (user_emp1_id, '(11) 99999-2222', 'São Paulo, SP', '1990-08-10', '2022-01-10',
pos_analista_rh_id, dept_rh_id, 'Mãe Maria', '(11) 88888-2222'),
    (user_emp2_id, '(11) 99999-3333', 'São Paulo, SP', '1988-12-05', '2021-11-20',
pos_dev_id, dept_ti_id, 'Pai Pedro', '(11) 88888-3333'),
    (user_emp3_id, '(11) 99999-4444', 'São Paulo, SP', '1992-03-25', '2023-02-01',
pos_designer_id, dept_ti_id, 'Irmã Ana', '(11) 88888-4444');
END $$;

-- Benefícios
INSERT INTO benefits (name, description, type, value) VALUES
('Plano de Saúde', 'Plano de saúde empresarial completo', 'health', 350.00),
('Plano Odontológico', 'Plano odontológico para funcionários', 'dental', 80.00),
('Vale Refeição', 'Vale refeição diário', 'meal', 25.00),
('Vale Transporte', 'Vale transporte mensal', 'transport', 150.00),
('Auxílio Educação', 'Auxílio para cursos e especializações', 'education', 500.00),
('Seguro de Vida', 'Seguro de vida empresarial', 'other', 50.00);

-- Treinamentos
INSERT INTO trainings (title, description, instructor, duration_hours,
max_participants, start_date, end_date, location, type, status) VALUES
('Segurança da Informação', 'Treinamento sobre boas práticas de segurança
digital', 'Carlos Security', 8, 20, '2024-02-15 09:00:00', '2024-02-15 17:00:00',
'Sala de Treinamento A', 'internal', 'scheduled'),
('Desenvolvimento React', 'Curso avançado de React e TypeScript', 'Ana Developer',
40, 15, '2024-03-01 09:00:00', '2024-03-05 17:00:00', 'Laboratório de Informática',
'internal', 'scheduled'),
('Gestão de Pessoas', 'Workshop sobre liderança e gestão de equipes', 'Maria
Leader', 16, 25, '2024-02-20 09:00:00', '2024-02-21 17:00:00', 'Auditório Principal',
'external', 'scheduled'),
('Excel Avançado', 'Treinamento de Excel para análise de dados', 'João Analyst', 12,
30, '2024-01-15 09:00:00', '2024-01-16 17:00:00', 'Sala de Treinamento B', 'internal',
'completed');

-- Inserir algumas participações em treinamentos
DO $$
DECLARE
    training_excel_id UUID;
    training_react_id UUID;
    user_emp1_id UUID;
    user_emp2_id UUID;
    user_emp3_id UUID;
BEGIN
    SELECT id INTO training_excel_id FROM trainings WHERE title = 'Excel Avançado';
    SELECT id INTO training_react_id FROM trainings WHERE title =
'Desenvolvimento React';
```

```
    SELECT id INTO user_emp1_id FROM users WHERE email =
'maria.santos@empresa.com';
    SELECT id INTO user_emp2_id FROM users WHERE email =
'pedro.oliveira@empresa.com';
    SELECT id INTO user_emp3_id FROM users WHERE email =
'ana.costa@empresa.com';

    INSERT INTO training_participations (training_id, user_id, status,
completion_date, score) VALUES
    (training_excel_id, user_emp1_id, 'completed', '2024-01-16', 8.5),
    (training_excel_id, user_emp2_id, 'completed', '2024-01-16', 9.0),
    (training_react_id, user_emp2_id, 'enrolled', NULL, NULL),
    (training_react_id, user_emp3_id, 'enrolled', NULL, NULL);
END $$;

-- Inserir algumas avaliações de exemplo
DO $$
DECLARE
    user_manager_id UUID;
    user_emp1_id UUID;
    user_emp2_id UUID;
BEGIN
    SELECT id INTO user_manager_id FROM users WHERE email =
'joao.silva@empresa.com';
    SELECT id INTO user_emp1_id FROM users WHERE email =
'maria.santos@empresa.com';
    SELECT id INTO user_emp2_id FROM users WHERE email =
'pedro.oliveira@empresa.com';

    INSERT INTO performance_evaluations (user_id, evaluator_id, period_start,
period_end, overall_score, goals_achievement, technical_skills, soft_skills,
comments, feedback, status) VALUES
    (user_emp1_id, user_manager_id, '2023-01-01', '2023-06-30', 8.5, 9.0, 8.0, 8.5,
'Excelente desempenho no período. Demonstrou grande comprometimento.',
'Continue desenvolvendo suas habilidades técnicas.', 'approved'),
    (user_emp2_id, user_manager_id, '2023-01-01', '2023-06-30', 9.0, 9.5, 9.0, 8.5,
'Desempenho excepcional. Superou todas as expectativas.', 'Considere assumir
mais responsabilidades de liderança.', 'approved');
END $$;

-- Inserir algumas solicitações de férias
DO $$
DECLARE
    user_emp1_id UUID;
    user_emp2_id UUID;
    user_manager_id UUID;
BEGIN
    SELECT id INTO user_emp1_id FROM users WHERE email =
'maria.santos@empresa.com';
    SELECT id INTO user_emp2_id FROM users WHERE email =
'pedro.oliveira@empresa.com';
    SELECT id INTO user_manager_id FROM users WHERE email =
```

```
'joao.silva@empresa.com';

    INSERT INTO vacations (user_id, start_date, end_date, days_requested, type,
status, approved_by, approved_at, comments) VALUES
    (user_emp1_id, '2024-07-01', '2024-07-15', 15, 'annual', 'approved',
user_manager_id, '2024-06-15 10:00:00', 'Férias aprovadas para período de verão'),
    (user_emp2_id, '2024-08-01', '2024-08-10', 10, 'annual', 'pending', NULL, NULL,
'Solicitação de férias para agosto'),
    (user_emp1_id, '2024-12-23', '2024-12-30', 8, 'annual', 'pending', NULL, NULL,
'Férias de fim de ano');
END $$;

-- Inserir algumas mensagens de chat de exemplo
DO $$
DECLARE
    user_admin_id UUID;
    user_emp1_id UUID;
BEGIN
    SELECT id INTO user_admin_id FROM users WHERE email =
'admin@hrmanagement.com';
    SELECT id INTO user_emp1_id FROM users WHERE email =
'maria.santos@empresa.com';

    INSERT INTO chat_messages (message, message_type, sender_id, recipient_id,
is_read) VALUES
    ('Olá! Como posso ajudá-lo hoje?', 'bot', NULL, user_emp1_id, false),
    ('Gostaria de saber sobre meus benefícios', 'user', user_emp1_id, NULL, true),
    ('Você pode consultar seus benefícios na seção "Benefícios" do sistema. Lá você
encontrará informações sobre plano de saúde, vale refeição e outros benefícios
disponíveis.', 'bot', NULL, user_emp1_id, false);
END $$;

-- Atribuir alguns benefícios aos usuários
DO $$
DECLARE
    benefit_saude_id UUID;
    benefit_refeicao_id UUID;
    benefit_transporte_id UUID;
    user_emp1_id UUID;
    user_emp2_id UUID;
    user_emp3_id UUID;
BEGIN
    SELECT id INTO benefit_saude_id FROM benefits WHERE name = 'Plano de
Saúde';
    SELECT id INTO benefit_refeicao_id FROM benefits WHERE name = 'Vale
Refeição';
    SELECT id INTO benefit_transporte_id FROM benefits WHERE name = 'Vale
Transporte';
    SELECT id INTO user_emp1_id FROM users WHERE email =
'maria.santos@empresa.com';
    SELECT id INTO user_emp2_id FROM users WHERE email =
'pedro.oliveira@empresa.com';
```

```
    SELECT id INTO user_emp3_id FROM users WHERE email =
'ana.costa@empresa.com';

    INSERT INTO user_benefits (user_id, benefit_id, start_date) VALUES
    (user_emp1_id, benefit_saude_id, '2022-01-10'),
    (user_emp1_id, benefit_refeicao_id, '2022-01-10'),
    (user_emp1_id, benefit_transporte_id, '2022-01-10'),
    (user_emp2_id, benefit_saude_id, '2021-11-20'),
    (user_emp2_id, benefit_refeicao_id, '2021-11-20'),
    (user_emp3_id, benefit_saude_id, '2023-02-01'),
    (user_emp3_id, benefit_refeicao_id, '2023-02-01');
END $$;

COMMIT;
```

## Credenciais de Acesso

Após executar os scripts, você pode acessar o sistema com:

- **Email**: admin@hrmanagement.com
- **Senha**: admin123

## Usuários de Demonstração

O script de dados de exemplo cria os seguintes usuários (todos com senha: admin123):

1. **admin@hrmanagement.com** - Administrador
2. **joao.silva@empresa.com** - Gerente de TI
3. **maria.santos@empresa.com** - Analista de RH
4. **pedro.oliveira@empresa.com** - Desenvolvedor Full Stack
5. **ana.costa@empresa.com** - Designer UX/UI
6. **carlos.ferreira@empresa.com** - Funcionário

## Verificação da Instalação

Após executar os scripts, você pode verificar se tudo foi criado corretamente:

```
 -- Verificar tabelas criadas
SELECT table_name FROM information_schema.tables
WHERE table_schema = 'public'
ORDER BY table_name;

-- Verificar usuários criados
SELECT email, name, role FROM users;
```

```sql
-- Verificar estatísticas do dashboard
SELECT * FROM dashboard_stats;

-- Verificar funcionários com detalhes
SELECT name, email, position_name, department_name
FROM employees_detailed
WHERE is_active = true;
```

## Próximos Passos

1. Execute os scripts SQL no Supabase
2. Configure as variáveis de ambiente do backend com a string de conexão do Supabase
3. Inicie o backend e frontend
4. Acesse o sistema com as credenciais fornecidas
5. Explore as funcionalidades e personalize conforme necessário

---

Scripts SQL - Sistema de Gestão de Pessoas

Versão 1.0 - Desenvolvido por Manus AI