**Centro Universitário UNISATC**

Engenharia de Software 3a fase – Banco de Dados II – Prof.Luciano B. Fernandes

**TRABALHO FINAL COM BASE EM METODOLOGIAS ATIVAS DE APRENDIZAGEM**

**Projeto de banco de dados para um sistema de Academia**

Diogo Dias de Abreu Alves + DiogoDiasAlves

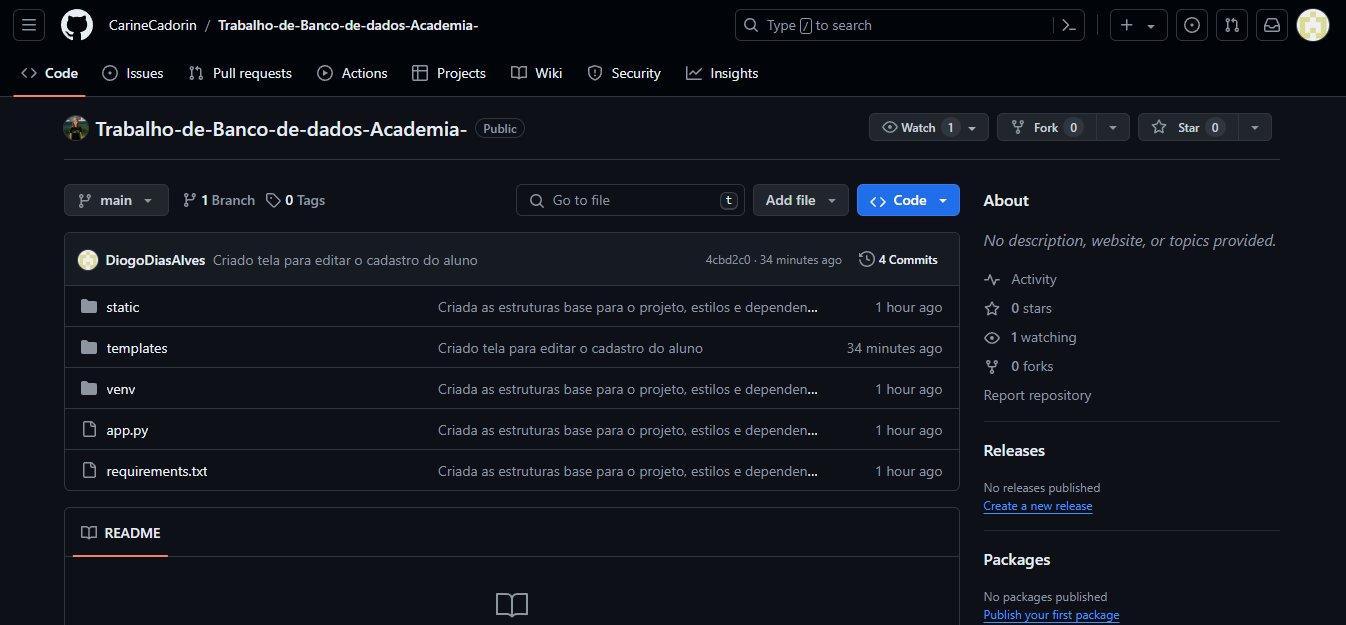
Vinicius Colombo + viniciuscolombo

Carine Cadorin + CarineCadorin

Criciúma, 15/06/2024

**URL do projeto no GitHub**

https://github.com/CarineCadorin/Trabalho-de-Banco-de-dados-Academia-



**Modelo ER Físico**

Diagrama

Descrição gerada automaticamente

**Dicionário de Dados**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Aluno | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados dos alunos da academia | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id\_aluno | | int | 1 - sem limite | NOT NULL | X |  | Código de identificação do aluno |
| nome | | VARCHAR(100) | 1 - 100 | NOT NULL |  |  | Nome do aluno |
| datanasc | | DATE |  | NOT NULL |  |  | Data de nascimento |
| endereco | | VARCHAR(255) | 1 - 255 | NOT NULL |  |  | Endereço residencial |
| telefone | | VARCHAR(20) | 10-20 | NOT NULL |  |  | Número de telefone |
| email | | VARCHAR(100) | 1 - 100 | NOT NULL |  |  | Endereço de email |
| datainicio | | DATE |  | NOT NULL |  |  | Data de início da associação |
| status | | VARCHAR(50) | 1-10 | NOT NULL |  |  | Status da associação |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx\_aluno\_nome | | Não | Sim | Sim | nome | | |
| idx\_aluno\_data\_inicio | | Não | Sim | Sim | dataInicio | | |
| idx\_aluno\_status | | Não | Sim | Sim | Status | | |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Tabela** | Inscricao | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados das inscrições dos alunos da academia | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id\_inscricao | | int | 1 – sem limite | NOT NULL | X |  | Código de identificação da inscrição |
| id\_aluno | | int | 1 – sem limite | NOT NULL |  | X | Identificador do aluno |
| id\_plano | | int | 1 – sem limite | NOT NULL |  | X | Identificador do plano |
| datainscricao | | DATE |  | NOT NULL |  |  | Data de inscrição |
| status | | VARCHAR(50) | 1-50 | NOT NULL |  |  | Status da inscrição |
| dataexpiracao | | DATE |  | NOT NULL |  |  | Data de expiração da inscrição |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Tabela** | Instrutor | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados dos instrutores da academia | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id\_instrutor | | int | 1 – sem limite | NOT NULL | X |  | Código de identificação do instrutor |
| nome | | VARCHAR(100) | 1 - 100 | NOT NULL |  |  | Nome do instrutor |
| especialidade | | VARCHAR(100) | 1 - 100 | NOT NULL |  |  | Especialidade do instrutor |
| certificacoes | | VARCHAR(255) | 1 - 255 | NOT NULL |  |  | Certificações do instrutor |
| email | | VARCHAR(100) | 1 - 100 | NOT NULL |  |  | Email do instrutor |
| anosexperiencia | | INT |  | NOT NULL |  |  | Anos de experiência do instrutor |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx\_instrutor\_especialidade | | Não | Sim | Sim | especialidade | | |
| idx\_instrutor\_nome | | Não | Sim | Sim | nome | | |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Tabela** | Plano\_Treinamento | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados dos planos de treinamentos da academia | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id\_plano | | int | 1 – sem limite | NOT NULL | X |  | Código de identificação do plano de treinamento |
| nome | | VARCHAR(100) | 1-100 | NOT NULL |  |  | Nome do plano de treinamento |
| descricao | | TEXT |  | NOT NULL |  |  | Descrição do plano de treinamento |
| duracao | | INT |  | NOT NULL |  |  | Duração do plano de treinamento |
| dataInicio | | DATE |  | NOT NULL |  |  | Data de início do plano de treinamento |
| dataFim | | DATE |  | NOT NULL |  |  | Data de fim do plano de treinamento |
| id\_equipamento | | INT | 1 – sem limite | NOT NULL |  | X | Identificador do equipamento |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Tabela** | Pagamento | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados dos pagementos da academia | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id\_pagamento | | INT | 1 – sem limite | NOT NULL | X |  | Código de identificação do instrutor |
| id\_aluno | | INT | 1 – sem limite | NOT NULL |  | X | Identificador do aluno |
| valor | | DECIMAL(10,2) |  | NOT NULL |  |  | Valor pago |
| data | | DATE |  | NOT NULL |  |  | Data do pagamento |
| metodo | | VARCHAR(50) | 1 - 50 | NOT NULL |  |  | Método de pagamento |
| status | | VARCHAR(50) | 1 - 50 | NOT NULL |  |  | Status do pagamento |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx\_pagamento\_aluno | | Não | Sim | Sim | aluno | | |
| idx\_pagamento\_data | | Não | Sim | Sim | data | | |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Tabela** | Avaliacao\_Fisica | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados de avaliação física dos alunos da academia | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id\_avaliacao | | INT | 1 – sem limite | NOT NULL | X |  | Identificador único da avaliação |
| id\_aluno | | INT | 1 – sem limite | NOT NULL |  | X | Identificador do aluno |
| id\_instrutor | | INT | 1 – sem limite | NOT NULL |  | X | Identificador do instrutor |
| data | | DATE |  | NOT NULL |  |  | Data da avaliação |
| peso | | DECIMAL(5,2) |  | NOT NULL |  |  | Peso do membro |
| altura | | DECIMAL(4,2) |  | NOT NULL |  |  | Altura do membro |
| gorduracorp | | DECIMAL(5,2) |  | NOT NULL |  |  | Percentual de gordura corporal |
| massamagra | | DECIMAL(5,2) |  | NOT NULL |  |  | Massa magra |
| pressao | | VARCHAR(20) |  | NOT NULL |  |  | Pressão arterial |
| observação | | TEXT |  | NOT NULL |  |  | Observações do instrutor |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Tabela** | Treino | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados dos treinos dos alunos da academia | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id\_aula | | INT | 1 – sem limite | NOT NULL | X |  | Identificador único da aula |
| id\_instrutor | | INT | 1 – sem limite | NOT NULL |  | X | Identificador do instrutor |
| id\_equipamento | | INT | 1 – sem limite | NOT NULL |  | X | Identificador do equipamento |
| horario | | TIME |  | NOT NULL |  |  | Horário da aula |
| duracao | | INT |  | NOT NULL |  |  | Duração em minutos |
| capacidade |  | INT |  | NOT NULL |  |  | Capacidade máxima |
| sala | | VARCHAR(100) | 1-100 | NOT NULL |  |  | Sala onde a aula é realizada |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx\_treino\_horario | | Não | Sim | Sim | horario | | |
| idx\_treino\_instrutor | | Não | Sim | Sim | instrutor | | |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Tabela** | Equipamento | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados dos equipamentos da academia | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id\_equipamento | | INT | 1 – sem limite | NOT NULL | X |  | Identificador único do equipamento |
| nome | | VARCHAR(100) | 1-100 | NOT NULL |  |  | Nome do equipamento |
| tipo | | VARCHAR(100) | 1-100 | NOT NULL |  |  | Tipo do equipamento |
| quantidade | | INT |  | NOT NULL |  |  | Quantidade disponível |
| dataaquisicao | | DATE |  | NOT NULL |  |  | Data de aquisição |
| estado |  | VARCHAR(50) | 1-50 | NOT NULL |  |  | Estado do equipamento |
| manutencaopendente | | BOOLEAN |  | NOT NULL |  |  | Indica se há manutenção pendente |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Tabela** | check\_in\_check\_out | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados de entrada e saída dos alunos da academia | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id\_checkin | | INT | 1 – sem limite | NOT NULL | X |  | Identificador único do equipamento |
| id\_aluno | | INT | 1 – sem limite | NOT NULL |  | X | Nome do equipamento |
| data | | DATA |  | NOT NULL |  |  | Tipo do equipamento |
| horaentrada | | TIME |  | NOT NULL |  |  | Quantidade disponível |
| horasaida | | TIME |  | NOT NULL |  |  | Data de aquisição |

**Script dos comandos DDL para criação do Banco de dados**

-- Database: academia

-- DROP DATABASE IF EXISTS academia;

CREATE DATABASE academia

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'Portuguese\_Brazil.1252'

LC\_CTYPE = 'Portuguese\_Brazil.1252'

LOCALE\_PROVIDER = 'libc'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

------------------------------------------------------------------------------------------------------------

-- Table: public.aluno

-- DROP TABLE IF EXISTS public.aluno;

CREATE TABLE IF NOT EXISTS public.aluno

(

id\_aluno integer NOT NULL DEFAULT nextval('aluno\_id\_aluno\_seq'::regclass),

nome character varying(100) COLLATE pg\_catalog."default" NOT NULL,

datanasc date NOT NULL,

endereco character varying(255) COLLATE pg\_catalog."default" NOT NULL,

telefone character varying(20) COLLATE pg\_catalog."default" NOT NULL,

email character varying(100) COLLATE pg\_catalog."default" NOT NULL,

datainicio date NOT NULL,

status character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT aluno\_pkey PRIMARY KEY (id\_aluno)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.aluno

OWNER to postgres;

-- Index: idx\_aluno\_data\_inicio

-- DROP INDEX IF EXISTS public.idx\_aluno\_data\_inicio;

CREATE INDEX IF NOT EXISTS idx\_aluno\_data\_inicio

ON public.aluno USING btree

(datainicio ASC NULLS LAST)

TABLESPACE pg\_default;

-- Index: idx\_aluno\_nome

-- DROP INDEX IF EXISTS public.idx\_aluno\_nome;

CREATE INDEX IF NOT EXISTS idx\_aluno\_nome

ON public.aluno USING btree

(nome COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

-- Index: idx\_aluno\_status

-- DROP INDEX IF EXISTS public.idx\_aluno\_status;

CREATE INDEX IF NOT EXISTS idx\_aluno\_status

ON public.aluno USING btree

(status COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

------------------------------------------------------------------------------------------------------------

-- Table: public.avaliacao\_fisica

-- DROP TABLE IF EXISTS public.avaliacao\_fisica;

CREATE TABLE IF NOT EXISTS public.avaliacao\_fisica

(

id\_avaliacao integer NOT NULL DEFAULT nextval('avaliacao\_fisica\_id\_avaliacao\_seq'::regclass),

id\_aluno integer NOT NULL,

id\_instrutor integer NOT NULL,

data date NOT NULL,

peso numeric(5,2) NOT NULL,

altura numeric(4,2) NOT NULL,

gorduracorp numeric(5,2) NOT NULL,

massamagra numeric(5,2) NOT NULL,

pressao character varying(20) COLLATE pg\_catalog."default" NOT NULL,

observacoes text COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT avaliacao\_fisica\_pkey PRIMARY KEY (id\_avaliacao),

CONSTRAINT avaliacao\_fisica\_id\_aluno\_fkey FOREIGN KEY (id\_aluno)

REFERENCES public.aluno (id\_aluno) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT avaliacao\_fisica\_id\_instrutor\_fkey FOREIGN KEY (id\_instrutor)

REFERENCES public.instrutor (id\_instrutor) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.avaliacao\_fisica

OWNER to postgres;

------------------------------------------------------------------------------------------------------------

-- Table: public.check\_in\_check\_out

-- DROP TABLE IF EXISTS public.check\_in\_check\_out;

CREATE TABLE IF NOT EXISTS public.check\_in\_check\_out

(

id\_checkin integer NOT NULL DEFAULT nextval('check\_in\_check\_out\_id\_checkin\_seq'::regclass),

id\_aluno integer NOT NULL,

data date NOT NULL,

horaentrada time without time zone NOT NULL,

horasaida time without time zone NOT NULL,

CONSTRAINT check\_in\_check\_out\_pkey PRIMARY KEY (id\_checkin),

CONSTRAINT check\_in\_check\_out\_id\_aluno\_fkey FOREIGN KEY (id\_aluno)

REFERENCES public.aluno (id\_aluno) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.check\_in\_check\_out

OWNER to postgres;

------------------------------------------------------------------------------------------------------------

-- Table: public.equipamento

-- DROP TABLE IF EXISTS public.equipamento;

CREATE TABLE IF NOT EXISTS public.equipamento

(

id\_equipamento integer NOT NULL DEFAULT nextval('equipamento\_id\_equipamento\_seq'::regclass),

nome character varying(100) COLLATE pg\_catalog."default" NOT NULL,

tipo character varying(100) COLLATE pg\_catalog."default" NOT NULL,

quantidade integer NOT NULL,

dataaquisicao date NOT NULL,

estado character varying(50) COLLATE pg\_catalog."default" NOT NULL,

manutencaopendente boolean NOT NULL,

CONSTRAINT equipamento\_pkey PRIMARY KEY (id\_equipamento)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.equipamento

OWNER to postgres;

------------------------------------------------------------------------------------------------------------

-- Table: public.inscricao

-- DROP TABLE IF EXISTS public.inscricao;

CREATE TABLE IF NOT EXISTS public.inscricao

(

id\_inscricao integer NOT NULL DEFAULT nextval('inscricao\_id\_inscricao\_seq'::regclass),

id\_aluno integer NOT NULL,

id\_plano integer NOT NULL,

datainscricao date NOT NULL,

status character varying(50) COLLATE pg\_catalog."default" NOT NULL,

dataexpiracao date NOT NULL,

CONSTRAINT inscricao\_pkey PRIMARY KEY (id\_inscricao),

CONSTRAINT inscricao\_id\_aluno\_fkey FOREIGN KEY (id\_aluno)

REFERENCES public.aluno (id\_aluno) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT inscricao\_id\_plano\_fkey FOREIGN KEY (id\_plano)

REFERENCES public.plano\_de\_treinamento (id\_plano) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.inscricao

OWNER to postgres;

-- Trigger: inscricao\_trigger

-- DROP TRIGGER IF EXISTS inscricao\_trigger ON public.inscricao;

CREATE OR REPLACE TRIGGER inscricao\_trigger

AFTER INSERT

ON public.inscricao

FOR EACH ROW

EXECUTE FUNCTION public.atualizar\_quantidade\_equipamento();

------------------------------------------------------------------------------------------------------------

-- Table: public.instrutor

-- DROP TABLE IF EXISTS public.instrutor;

CREATE TABLE IF NOT EXISTS public.instrutor

(

id\_instrutor integer NOT NULL DEFAULT nextval('instrutor\_id\_instrutor\_seq'::regclass),

nome character varying(100) COLLATE pg\_catalog."default" NOT NULL,

especialidade character varying(100) COLLATE pg\_catalog."default" NOT NULL,

certificacoes character varying(255) COLLATE pg\_catalog."default" NOT NULL,

anosexperiencia integer NOT NULL,

email character varying(100) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT instrutor\_pkey PRIMARY KEY (id\_instrutor)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.instrutor

OWNER to postgres;

-- Index: idx\_instrutor\_especialidade

-- DROP INDEX IF EXISTS public.idx\_instrutor\_especialidade;

CREATE INDEX IF NOT EXISTS idx\_instrutor\_especialidade

ON public.instrutor USING btree

(especialidade COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

-- Index: idx\_instrutor\_nome

-- DROP INDEX IF EXISTS public.idx\_instrutor\_nome;

CREATE INDEX IF NOT EXISTS idx\_instrutor\_nome

ON public.instrutor USING btree

(nome COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

------------------------------------------------------------------------------------------------------------

-- Table: public.pagamento

-- DROP TABLE IF EXISTS public.pagamento;

CREATE TABLE IF NOT EXISTS public.pagamento

(

id\_pagamento integer NOT NULL DEFAULT nextval('pagamento\_id\_pagamento\_seq'::regclass),

id\_aluno integer NOT NULL,

valor numeric(10,2) NOT NULL,

data date NOT NULL,

metodo character varying(50) COLLATE pg\_catalog."default" NOT NULL,

status character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT pagamento\_pkey PRIMARY KEY (id\_pagamento),

CONSTRAINT pagamento\_id\_aluno\_fkey FOREIGN KEY (id\_aluno)

REFERENCES public.aluno (id\_aluno) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.pagamento

OWNER to postgres;

-- Index: idx\_pagamento\_aluno

-- DROP INDEX IF EXISTS public.idx\_pagamento\_aluno;

CREATE INDEX IF NOT EXISTS idx\_pagamento\_aluno

ON public.pagamento USING btree

(id\_aluno ASC NULLS LAST)

TABLESPACE pg\_default;

-- Index: idx\_pagamento\_data

-- DROP INDEX IF EXISTS public.idx\_pagamento\_data;

CREATE INDEX IF NOT EXISTS idx\_pagamento\_data

ON public.pagamento USING btree

(data ASC NULLS LAST)

TABLESPACE pg\_default;

------------------------------------------------------------------------------------------------------------

-- Table: public.plano\_de\_treinamento

-- DROP TABLE IF EXISTS public.plano\_de\_treinamento;

CREATE TABLE IF NOT EXISTS public.plano\_de\_treinamento

(

id\_plano integer NOT NULL DEFAULT nextval('plano\_de\_treinamento\_id\_plano\_seq'::regclass),

nome character varying(100) COLLATE pg\_catalog."default" NOT NULL,

descricao text COLLATE pg\_catalog."default" NOT NULL,

duracao integer NOT NULL,

datainicio date NOT NULL,

datafim date NOT NULL,

id\_equipamento integer NOT NULL,

CONSTRAINT plano\_de\_treinamento\_pkey PRIMARY KEY (id\_plano),

CONSTRAINT plano\_de\_treinamento\_id\_equipamento\_fkey FOREIGN KEY (id\_equipamento)

REFERENCES public.equipamento (id\_equipamento) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.plano\_de\_treinamento

OWNER to postgres;

------------------------------------------------------------------------------------------------------------

-- Table: public.treino

-- DROP TABLE IF EXISTS public.treino;

CREATE TABLE IF NOT EXISTS public.treino

(

id\_aula integer NOT NULL DEFAULT nextval('treino\_id\_aula\_seq'::regclass),

id\_instrutor integer NOT NULL,

id\_equipamento integer NOT NULL,

horario time without time zone NOT NULL,

duracao integer NOT NULL,

capacidade integer NOT NULL,

sala character varying(100) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT treino\_pkey PRIMARY KEY (id\_aula),

CONSTRAINT treino\_id\_equipamento\_fkey FOREIGN KEY (id\_equipamento)

REFERENCES public.equipamento (id\_equipamento) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT treino\_id\_instrutor\_fkey FOREIGN KEY (id\_instrutor)

REFERENCES public.instrutor (id\_instrutor) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.treino

OWNER to postgres;

-- Index: idx\_treino\_horario

-- DROP INDEX IF EXISTS public.idx\_treino\_horario;

CREATE INDEX IF NOT EXISTS idx\_treino\_horario

ON public.treino USING btree

(horario ASC NULLS LAST)

TABLESPACE pg\_default;

-- Index: idx\_treino\_instrutor

-- DROP INDEX IF EXISTS public.idx\_treino\_instrutor;

CREATE INDEX IF NOT EXISTS idx\_treino\_instrutor

ON public.treino USING btree

(id\_instrutor ASC NULLS LAST)

TABLESPACE pg\_default;

------------------------------------------------------------------------------------------------------------

-- View: public.vw\_membros\_ativos\_por\_plano

-- DROP VIEW public.vw\_membros\_ativos\_por\_plano;

CREATE OR REPLACE VIEW public.vw\_membros\_ativos\_por\_plano

AS

SELECT p.nome AS nome\_plano,

count(i.id\_inscricao) AS quantidade\_membros\_ativos

FROM inscricao i

JOIN plano\_de\_treinamento p ON i.id\_plano = p.id\_plano

WHERE i.status::text = 'Ativo'::text

GROUP BY p.nome;

ALTER TABLE public.vw\_membros\_ativos\_por\_plano

OWNER TO postgres;

------------------------------------------------------------------------------------------------------------

-- View: public.vw\_membros\_por\_instrutor

-- DROP VIEW public.vw\_membros\_por\_instrutor;

CREATE OR REPLACE VIEW public.vw\_membros\_por\_instrutor

AS

SELECT i.nome AS nome\_instrutor,

count(DISTINCT c.id\_aluno) AS quantidade\_membros

FROM instrutor i

LEFT JOIN treino t ON i.id\_instrutor = t.id\_instrutor

LEFT JOIN check\_in\_check\_out c ON t.id\_aula = c.id\_checkin

GROUP BY i.nome;

ALTER TABLE public.vw\_membros\_por\_instrutor

OWNER TO postgres;

------------------------------------------------------------------------------------------------------------

-- View: public.vw\_receita\_mensal

-- DROP VIEW public.vw\_receita\_mensal;

CREATE OR REPLACE VIEW public.vw\_receita\_mensal

AS

SELECT date\_trunc('month'::text, data::timestamp with time zone) AS ano\_mes,

sum(valor) AS receita\_total

FROM pagamento p

WHERE data >= (CURRENT\_DATE - '1 year'::interval)

GROUP BY (date\_trunc('month'::text, data::timestamp with time zone));

ALTER TABLE public.vw\_receita\_mensal

OWNER TO postgres;

------------------------------------------------------------------------------------------------------------

--Esta função insere um novo aluno na tabela Aluno e retorna o ID do aluno inserido.

CREATE OR REPLACE FUNCTION inserir\_aluno(

p\_id\_aluno INT,

p\_nome VARCHAR(100),

p\_data\_nasc DATE,

p\_endereco VARCHAR(255),

p\_telefone VARCHAR(20),

p\_email VARCHAR(100),

p\_data\_inicio DATE,

p\_status VARCHAR(50)

) RETURNS INT AS $$

BEGIN

-- Insere um novo aluno com o ID especificado

INSERT INTO Aluno (ID\_Aluno, Nome, DataNasc, Endereco, Telefone, Email, DataInicio, Status)

VALUES (p\_id\_aluno, p\_nome, p\_data\_nasc, p\_endereco, p\_telefone, p\_email, p\_data\_inicio, p\_status);

RETURN p\_id\_aluno;

END;

$$ LANGUAGE plpgsql;

------------------------------------------------------------------------------------------------------------

--Esta função calcula o IMC de um aluno com base no peso e altura registrados em uma avaliação física.

CREATE OR REPLACE FUNCTION calcular\_imc(

p\_peso DECIMAL(5, 2),

p\_altura DECIMAL(4, 2)

) RETURNS DECIMAL(5, 2) AS $$

DECLARE

imc DECIMAL(5, 2);

BEGIN

imc := p\_peso / (p\_altura \* p\_altura);

RETURN imc;

END;

$$ LANGUAGE plpgsql;

------------------------------------------------------------------------------------------------------------

--Esta função atualiza o status de pagamento de um aluno na tabela Pagamento com base no ID do aluno e na data do pagamento.

CREATE OR REPLACE FUNCTION atualizar\_status\_pagamento(

p\_id\_aluno INT,

p\_data\_pagamento DATE,

p\_status VARCHAR(50)

) RETURNS VOID AS $$

BEGIN

UPDATE Pagamento

SET Status = p\_status

WHERE ID\_Aluno = p\_id\_aluno

AND Data = p\_data\_pagamento;

END;

$$ LANGUAGE plpgsql;

**Script que popula as tabelas do Banco de dados**

-- Populando a tabela Equipamento

INSERT INTO Equipamento (ID\_Equipamento, Nome, Tipo, Quantidade, DataAquisicao, Estado, ManutencaoPendente)

VALUES

(1, 'Esteira', 'Cardio', 10, '2022-01-01', 'Bom', false),

(2, 'Bicicleta', 'Cardio', 5, '2022-02-01', 'Bom', false),

(3, 'Halteres', 'Musculação', 20, '2022-03-01', 'Bom', false),

(4, 'Máquina de Remo', 'Cardio', 2, '2022-04-01', 'Ruim', true),

(5, 'Leg Press', 'Musculação', 1, '2022-05-01', 'Bom', false),

(6, 'Elíptico', 'Cardio', 3, '2022-06-01', 'Bom', false),

(7, 'Cadeira Extensora', 'Musculação', 4, '2022-07-01', 'Bom', false),

(8, 'Puxador', 'Musculação', 2, '2022-08-01', 'Ruim', true),

(9, 'Bicicleta Horizontal', 'Cardio', 3, '2022-09-01', 'Bom', false),

(10, 'Esteira Curva', 'Cardio', 2, '2022-10-01', 'Ruim', true),

(11, 'Smith Machine', 'Musculação', 1, '2022-11-01', 'Bom', false),

(12, 'Banco Supino', 'Musculação', 5, '2022-12-01', 'Bom', false),

(13, 'Step', 'Cardio', 10, '2023-01-01', 'Bom', false),

(14, 'Corda de Pular', 'Cardio', 15, '2023-02-01', 'Bom', false),

(15, 'Kettlebell', 'Musculação', 25, '2023-03-01', 'Bom', false);

-- Populando a tabela Aluno

INSERT INTO Aluno (ID\_Aluno, Nome, DataNasc, Endereco, Telefone, Email, DataInicio, Status)

VALUES

(1, 'Carlos Silva', '1990-05-10', 'Rua A, 123', '1111-1111', 'carlos@email.com', '2023-01-01', 'Ativo'),

(2, 'Maria Oliveira', '1985-08-22', 'Rua B, 456', '2222-2222', 'maria@email.com', '2023-02-01', 'Ativo'),

(3, 'João Souza', '1992-07-15', 'Rua C, 789', '3333-3333', 'joao@email.com', '2023-03-01', 'Inativo'),

(4, 'Ana Santos', '1988-11-30', 'Rua D, 101', '4444-4444', 'ana@email.com', '2023-04-01', 'Ativo'),

(5, 'Pedro Lima', '1995-12-12', 'Rua E, 202', '5555-5555', 'pedro@email.com', '2023-05-01', 'Ativo'),

(6, 'Paula Mendes', '1991-09-20', 'Rua F, 303', '6666-6666', 'paula@email.com', '2023-06-01', 'Inativo'),

(7, 'Ricardo Alves', '1983-04-18', 'Rua G, 404', '7777-7777', 'ricardo@email.com', '2023-07-01', 'Ativo'),

(8, 'Fernanda Costa', '1987-02-05', 'Rua H, 505', '8888-8888', 'fernanda@email.com', '2023-08-01', 'Ativo'),

(9, 'Bruno Martins', '1993-06-25', 'Rua I, 606', '9999-9999', 'bruno@email.com', '2023-09-01', 'Ativo'),

(10, 'Renata Pereira', '1989-03-14', 'Rua J, 707', '1010-1010', 'renata@email.com', '2023-10-01', 'Ativo'),

(11, 'Thiago Ramos', '1986-12-01', 'Rua K, 808', '1111-1112', 'thiago@email.com', '2023-11-01', 'Ativo'),

(12, 'Juliana Barbosa', '1994-01-19', 'Rua L, 909', '1212-1212', 'juliana@email.com', '2023-12-01', 'Inativo'),

(13, 'Roberto Ferreira', '1990-05-17', 'Rua M, 111', '1313-1313', 'roberto@email.com', '2023-01-01', 'Ativo'),

(14, 'Débora Nascimento', '1985-08-27', 'Rua N, 222', '1414-1414', 'debora@email.com', '2023-02-01', 'Ativo'),

(15, 'Leonardo Pinto', '1992-07-09', 'Rua O, 333', '1515-1515', 'leonardo@email.com', '2023-03-01', 'Ativo');

-- Populando a tabela Instrutor

INSERT INTO Instrutor (ID\_Instrutor, Nome, Especialidade, Certificacoes, AnosExperiencia, Email)

VALUES

(1, 'Alice', 'Cardio', 'Certificação A', 5, 'alice@email.com'),

(2, 'Bruno', 'Musculação', 'Certificação B', 7, 'bruno@email.com'),

(3, 'Carlos', 'Yoga', 'Certificação C', 10, 'carlos@email.com'),

(4, 'Daniela', 'Pilates', 'Certificação D', 8, 'daniela@email.com'),

(5, 'Eduardo', 'Crossfit', 'Certificação E', 6, 'eduardo@email.com'),

(6, 'Fernanda', 'Zumba', 'Certificação F', 9, 'fernanda@email.com'),

(7, 'Gabriel', 'Cardio', 'Certificação G', 4, 'gabriel@email.com'),

(8, 'Helena', 'Musculação', 'Certificação H', 5, 'helena@email.com'),

(9, 'Igor', 'Yoga', 'Certificação I', 7, 'igor@email.com'),

(10, 'Julia', 'Pilates', 'Certificação J', 10, 'julia@email.com'),

(11, 'Kleber', 'Crossfit', 'Certificação K', 8, 'kleber@email.com'),

(12, 'Larissa', 'Zumba', 'Certificação L', 6, 'larissa@email.com'),

(13, 'Marcos', 'Cardio', 'Certificação M', 9, 'marcos@email.com'),

(14, 'Natalia', 'Musculação', 'Certificação N', 4, 'natalia@email.com'),

(15, 'Otavio', 'Yoga', 'Certificação O', 5, 'otavio@email.com');

-- Populando a tabela Plano\_de\_Treinamento

INSERT INTO Plano\_de\_Treinamento (ID\_Plano, Nome, Descricao, Duracao, DataInicio, DataFim, ID\_Equipamento)

VALUES

(1, 'Plano Básico', 'Acesso livre', 30, '2023-01-01', '2023-12-31', 1),

(2, 'Plano Premium', 'Acesso + Personal Trainer', 30, '2023-01-01', '2023-12-31', 2),

(3, 'Plano VIP', 'Acesso total', 30, '2023-01-01', '2023-12-31', 3),

(4, 'Plano Família', 'Acesso para família', 30, '2023-01-01', '2023-12-31', 4),

(5, 'Plano Estudante', 'Acesso para estudantes', 30, '2023-01-01', '2023-12-31', 5),

(6, 'Plano Senior', 'Acesso para idosos', 30, '2023-01-01', '2023-12-31', 6),

(7, 'Plano Corporativo', 'Acesso para empresas', 30, '2023-01-01', '2023-12-31', 7),

(8, 'Plano Fit', 'Acesso + Nutricionista', 30, '2023-01-01', '2023-12-31', 8),

(9, 'Plano Yoga', 'Acesso às aulas de Yoga', 30, '2023-01-01', '2023-12-31', 9),

(10, 'Plano Pilates', 'Acesso às aulas de Pilates', 30, '2023-01-01', '2023-12-31', 10),

(11, 'Plano Crossfit', 'Acesso às aulas de Crossfit', 30, '2023-01-01', '2023-12-31', 11),

(12, 'Plano Zumba', 'Acesso às aulas de Zumba', 30, '2023-01-01', '2023-12-31', 12),

(13, 'Plano Cardio', 'Acesso às aulas de Cardio', 30, '2023-01-01', '2023-12-31', 13),

(14, 'Plano Musculação', 'Acesso às aulas de Musculação', 30, '2023-01-01', '2023-12-31', 14),

(15, 'Plano Personal', 'Acesso com Personal Trainer', 30, '2023-01-01', '2023-12-31', 15);

-- Populando a tabela Inscrição

INSERT INTO Inscricao (ID\_Inscricao, ID\_Aluno, ID\_Plano, DataInscricao, Status, DataExpiracao)

VALUES

(1, 1, 1, '2023-01-01', 'Ativo', '2023-12-31'),

(2, 2, 2, '2023-02-01', 'Ativo', '2023-12-31'),

(3, 3, 3, '2023-03-01', 'Inativo', '2023-12-31'),

(4, 4, 4, '2023-04-01', 'Ativo', '2023-12-31'),

(5, 5, 5, '2023-05-01', 'Ativo', '2023-12-31'),

(6, 6, 6, '2023-06-01', 'Inativo', '2023-12-31'),

(7, 7, 7, '2023-07-01', 'Ativo', '2023-12-31'),

(8, 8, 8, '2023-08-01', 'Ativo', '2023-12-31'),

(9, 9, 9, '2023-09-01', 'Ativo', '2023-12-31'),

(10, 10, 10, '2023-10-01', 'Ativo', '2023-12-31'),

(11, 11, 11, '2023-11-01', 'Ativo', '2023-12-31'),

(12, 12, 12, '2023-12-01', 'Inativo', '2023-12-31'),

(13, 13, 13, '2023-01-01', 'Ativo', '2023-12-31'),

(14, 14, 14, '2023-02-01', 'Ativo', '2023-12-31'),

(15, 15, 15, '2023-03-01', 'Ativo', '2023-12-31');

-- Populando a tabela Avaliacao\_Fisica

INSERT INTO Avaliacao\_Fisica (ID\_Avaliacao, ID\_Aluno, ID\_Instrutor, Data, Peso, Altura, GorduraCorp, MassaMagra, Pressao, Observacoes)

VALUES

(1, 1, 1, '2023-01-01', 70.0, 1.75, 20.0, 50.0, '120/80', 'Nenhuma'),

(2, 2, 2, '2023-02-01', 60.0, 1.65, 22.0, 45.0, '120/80', 'Nenhuma'),

(3, 3, 3, '2023-03-01', 80.0, 1.80, 25.0, 55.0, '130/85', 'Nenhuma'),

(4, 4, 4, '2023-04-01', 90.0, 1.85, 30.0, 60.0, '140/90', 'Nenhuma'),

(5, 5, 5, '2023-05-01', 75.0, 1.70, 18.0, 52.0, '120/80', 'Nenhuma'),

(6, 6, 6, '2023-06-01', 65.0, 1.60, 20.0, 48.0, '110/70', 'Nenhuma'),

(7, 7, 7, '2023-07-01', 85.0, 1.90, 28.0, 58.0, '130/85', 'Nenhuma'),

(8, 8, 8, '2023-08-01', 55.0, 1.55, 25.0, 40.0, '120/80', 'Nenhuma'),

(9, 9, 9, '2023-09-01', 95.0, 1.95, 35.0, 65.0, '140/90', 'Nenhuma'),

(10, 10, 10, '2023-10-01', 70.0, 1.75, 20.0, 50.0, '120/80', 'Nenhuma'),

(11, 11, 11, '2023-11-01', 60.0, 1.65, 22.0, 45.0, '120/80', 'Nenhuma'),

(12, 12, 12, '2023-12-01', 80.0, 1.80, 25.0, 55.0, '130/85', 'Nenhuma'),

(13, 13, 13, '2023-01-01', 90.0, 1.85, 30.0, 60.0, '140/90', 'Nenhuma'),

(14, 14, 14, '2023-02-01', 75.0, 1.70, 18.0, 52.0, '120/80', 'Nenhuma'),

(15, 15, 15, '2023-03-01', 65.0, 1.60, 20.0, 48.0, '110/70', 'Nenhuma');

-- Populando a tabela Treino

INSERT INTO Treino (ID\_Aula, ID\_Instrutor, ID\_Equipamento, Horario, Duracao, Capacidade, Sala)

VALUES

(1, 1, 1, '08:00:00', 60, 10, 'Sala 1'),

(2, 2, 2, '09:00:00', 60, 15, 'Sala 2'),

(3, 3, 3, '10:00:00', 60, 20, 'Sala 3'),

(4, 4, 4, '11:00:00', 60, 25, 'Sala 4'),

(5, 5, 5, '12:00:00', 60, 10, 'Sala 5'),

(6, 6, 6, '13:00:00', 60, 15, 'Sala 6'),

(7, 7, 7, '14:00:00', 60, 20, 'Sala 7'),

(8, 8, 8, '15:00:00', 60, 25, 'Sala 8'),

(9, 9, 9, '16:00:00', 60, 10, 'Sala 9'),

(10, 10, 10,'17:00:00', 60, 15, 'Sala 10'),

(11, 11, 11,'18:00:00', 60, 20, 'Sala 11'),

(12, 12, 12,'19:00:00', 60, 25, 'Sala 12'),

(13, 13, 13,'20:00:00', 60, 10, 'Sala 13'),

(14, 14, 14,'21:00:00', 60, 15, 'Sala 14'),

(15, 15, 15,'22:00:00', 60, 20, 'Sala 15');

-- Populando a tabela Check-in/Check-out

INSERT INTO Check\_in\_Check\_out (ID\_Checkin, ID\_Aluno, Data, HoraEntrada, HoraSaida)

VALUES

(1, 1, '2023-01-01', '08:00:00', '09:00:00'),

(2, 2, '2023-02-01', '09:00:00', '10:00:00'),

(3, 3, '2023-03-01', '10:00:00', '11:00:00'),

(4, 4, '2023-04-01', '11:00:00', '12:00:00'),

(5, 5, '2023-05-01', '12:00:00', '13:00:00'),

(6, 6, '2023-06-01', '13:00:00', '14:00:00'),

(7, 7, '2023-07-01', '14:00:00', '15:00:00'),

(8, 8, '2023-08-01', '15:00:00', '16:00:00'),

(9, 9, '2023-09-01', '16:00:00', '17:00:00'),

(10, 10, '2023-10-01', '17:00:00', '18:00:00'),

(11, 11, '2023-11-01', '18:00:00', '19:00:00'),

(12, 12, '2023-12-01', '19:00:00', '20:00:00'),

(13, 13, '2023-01-01', '08:00:00', '09:00:00'),

(14, 14, '2023-02-01', '09:00:00', '10:00:00'),

(15, 15, '2023-03-01', '10:00:00', '11:00:00');

-- Populando a tabela Pagamento

INSERT INTO Pagamento (ID\_Pagamento, ID\_Aluno, Valor, Data, Metodo, Status)

VALUES

(1, 1, 100.0, '2023-01-01', 'Cartão', 'Pago'),

(2, 2, 200.0, '2023-02-01', 'Cartão', 'Pago'),

(3, 3, 300.0, '2023-03-01', 'Cartão', 'Pendente'),

(4, 4, 400.0, '2023-04-01', 'Cartão', 'Pago'),

(5, 5, 500.0, '2023-05-01', 'Cartão', 'Pago'),

(6, 6, 600.0, '2023-06-01', 'Cartão', 'Pendente'),

(7, 7, 700.0, '2023-07-01', 'Cartão', 'Pago'),

(8, 8, 800.0, '2023-08-01', 'Cartão', 'Pago'),

(9, 9, 900.0, '2023-09-01', 'Cartão', 'Pago'),

(10, 10, 1000.0, '2023-10-01', 'Cartão', 'Pago'),

(11, 11, 1100.0, '2023-11-01', 'Cartão', 'Pago'),

(12, 12, 1200.0, '2023-12-01', 'Cartão', 'Pendente'),

(13, 13, 1300.0, '2023-01-01', 'Cartão', 'Pago'),

(14, 14, 1400.0, '2023-02-01', 'Cartão', 'Pago'),

(15, 15, 1500.0, '2023-03-01', 'Cartão', 'Pago');

**Principais consultas mapeadas baseadas em regras de negócio (mínimo 4)**

--Quantos membros ativos estão matriculados em cada tipo de plano de assinatura?

SELECT

p.Nome AS Plano,

COUNT(i.ID\_Aluno) AS MembrosAtivos

FROM

Inscricao i

JOIN Plano\_de\_Treinamento p ON i.ID\_Plano = p.ID\_Plano

WHERE

i.Status = 'Ativo'

GROUP BY

p.Nome;

----------------------------------------------------------

--Qual é a receita total gerada por mês no último ano?

SELECT

DATE\_TRUNC('month', P.Data) AS Ano\_Mes,

SUM(P.Valor) AS Receita\_Total

FROM

Pagamento P

WHERE

P.Data >= CURRENT\_DATE - INTERVAL '1 year'

GROUP BY

Ano\_Mes;

------------------------------------------------------------

--Quais são as aulas mais populares e quantos membros estão inscritos em cada uma?

SELECT

T.ID\_Aula,

COUNT(C.ID\_Checkin) AS Quantidade\_Membros,

T.Sala

FROM

Treino T

LEFT JOIN

Check\_in\_Check\_out C ON T.ID\_Aula = C.ID\_Checkin

GROUP BY

T.ID\_Aula, T.Sala

ORDER BY

Quantidade\_Membros DESC;

-----------------------------------------------------------

--Quantos membros cada treinador está treinando atualmente?

SELECT

I.Nome AS Nome\_Instrutor,

COUNT(DISTINCT C.ID\_Aluno) AS Quantidade\_Membros

FROM

Instrutor I

LEFT JOIN

Treino T ON I.ID\_Instrutor = T.ID\_Instrutor

LEFT JOIN

Check\_in\_Check\_out C ON T.ID\_Aula = C.ID\_Checkin

GROUP BY

I.Nome;