**Centro Universitário UNISATC**

Engenharia de Software 3a fase – Banco de Dados II – Prof.Jorge Luiz da Silva

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

**Projeto de banco de dados para um sistema de Gestão de Estacionamento**

Alex Farias De Abreu Nabo - @Alex-Farias

Icaro De Oliveira - @ynd-icaro

Jean Lucas Preis - @JeanPrei

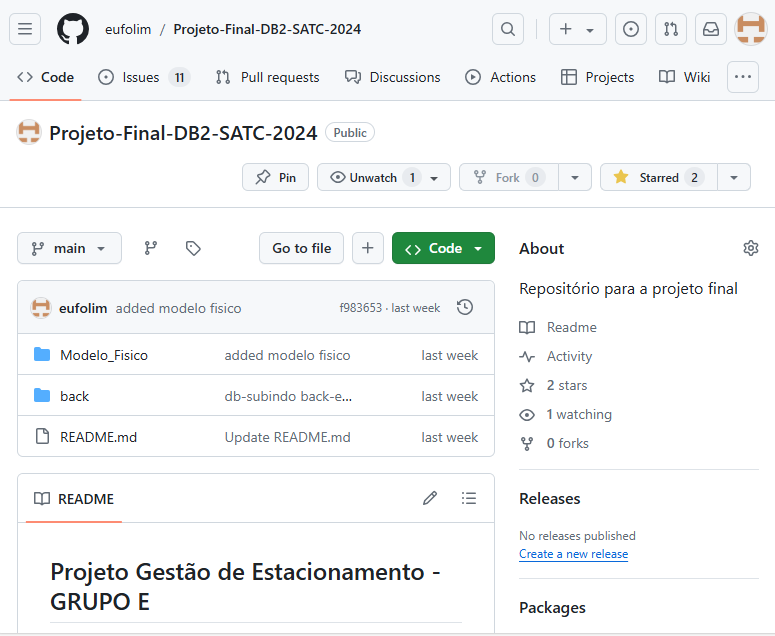
Luiz Antônio Frey Cristiano - @eufolim

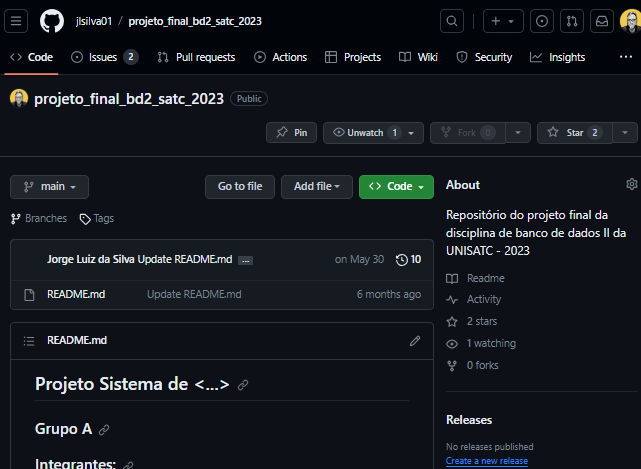
Matheus Araildi - @Araldi42<br>

Criciúma, 26/06/2024

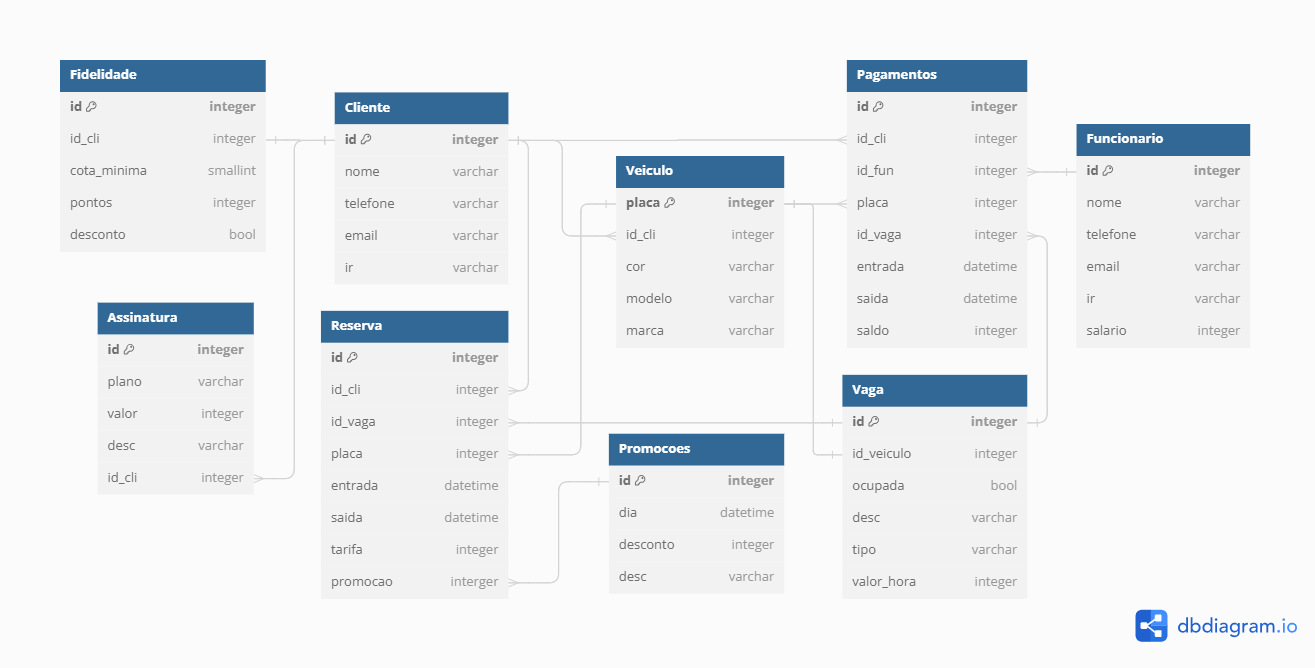
**URL do projeto no GitHub**

<https://github.com/eufolim/Projeto-Final-DB2-SATC-2024>

****

****

**Modelo ER Físico**

****

**Dicionário de Dados**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Vaga | | | | | | |
| **Descrição** | Tabela para guardar informações sobre cada vaga e seu estado atual | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id | | int | 1 – 2,147,483,647 | NOT NULL | X |  | Código de identificação de vaga |
| Id\_veiculo | | char | 1 - 20 | NOT NULL |  | X | Referencia a placa do veiculo |
| ocupada | | bool | 0 - 1 | NOT NULL |  |  | Estado da vaga |
| desc | | varchar | 1 - 255 | NOT NULL |  |  | Descrição da vaga |
| tipo | | varchar | 1 - 50 | NOT NULL |  |  | Tipo da vaga (para idoso, cadeirante, ...) |
| Valor\_hora | | int | 1 – 2,147,483,647 | NOT NULL |  |  | Valor a ser pago por hora estacionado |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
|  | |  |  |  |  | | |
|  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Veiculo | | | | | | |
| **Descrição** | Tabela para armazenar informações dos veículos, um cliente pode ter um ou mais veículos | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| placa | | char | 1 - 20 | NOT NULL | X |  | Placa do carro, usada como identificador |
| Id\_cli | | int | 1 - 2,147,483,647 | NOT NULL |  | X | Referencia o cliente dono do veiculo |
| cor | | varchar | 1 - 50 | NOT NULL |  |  | Cor do caro |
| modelo | | varchar | 1 - 50 | NOT NULL |  |  | Modelo do carro |
| marca | | varchar | 1 - 50 | NOT NULL |  |  | Marca do carro |
|  | |  |  |  |  |  |  |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
|  | |  |  |  |  | | |
|  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Cliente | | | | | | |
| **Descrição** | Tabela para armazenar informações dos clientes | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id | | int | 1 – 2,147,483,647 | NOT NULL | X |  | Identificador do cliente |
| nome | | varchar | 1 - 110 | NOT NULL |  |  | Nome do cliente |
| telefone | | int | 8 - 20 | NOT NULL |  |  | Numero de telefone do cliente |
| email | | varchar | 1 - 255 | NOT NULL |  |  | Email do cliente |
| ir | | varchar | 1 - 20 | NOT NULL |  |  | CPF ou RG do cliente |
|  | |  |  |  |  |  |  |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
|  | |  |  |  |  | | |
|  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Funcionario | | | | | | |
| **Descrição** | Tabela para armazenar informações dos funcionarios | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id | | int | 1 – 2,147,483,647 | NOT NULL | X |  | Identificador do funcionário |
| nome | | varchar | 1 - 110 | NOT NULL |  |  | Nome do funcionário |
| telefone | | int | 8 - 20 | NOT NULL |  |  | Numero de telefone do funcionário |
| email | | varchar | 1 - 255 | NOT NULL |  |  | Email do funcionário |
| ir | | varchar | 1 - 20 | NOT NULL |  |  | CPR ou RG do funcionário |
| salario | | int | 1 - 2,147,483,647 | NOT NULL |  |  | Salário do funcionário |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
|  | |  |  |  |  | | |
|  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Assinatura | | | | | | |
| **Descrição** | Tabela para armazenar assinatura mensal/anual do cliente | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id | | int | 1 – 2,147,483,647 | NOT NULL | X |  | Identificador da assinatura |
| plano | | varchar | 1 - 50 | NOT NULL |  |  | Plano de assinatura (mensal, anual, ...) |
| valor | | int | 1 - 2,147,483,647 | NOT NULL |  |  | Valor a ser pago pela assinatura |
| desck | | varchar | 1 - 255 | NOT NULL |  |  | Descrição da assinatura |
| Id\_cli | | int | 1 - 2,147,483,647 | NOT NULL |  | X | Referencia |
|  | |  |  |  |  |  |  |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
|  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Pagamentos | | | | | | |
| **Descrição** | Tabela para armazenar informações sobre o pagamento/tíquete | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id | | int | 1 – 2,147,483,647 | NOT NULL | X |  | Identificador da transação |
| Id\_cli | | int | 1 – 2,147,483,647 | NOT NULL |  | X | Referencia o cliente |
| Id\_fun | | int | 1 - 2,147,483,647 | NOT NULL |  | X | Referencia o funcionário responsável pelo atendimento |
| placa | | char | 1 - 20 | NOT NULL |  | X | Referencia a placa do veiculo |
| Id\_vaga | | int | 1 - 2,147,483,647 | NOT NULL |  | X | Referencia a vaga ocupada |
| entrada | | datetime | 1000-01-01 00:00:00 - 9999-12-31 23:59:59 |  |  |  | Armazena o horário de entrada do cliente após receber o tíquete |
| saida | | datetime | 1000-01-01 00:00:00 - 9999-12-31 23:59:59 |  |  |  | Armazena o horário de saída do cliente após entregar o tíquete |
| saldo | | int | 1 - 2,147,483,647 |  |  |  | Valor a ser pago |
| Total\_horas | | int | 1 - 2,147,483,647 |  |  |  | Total de horas estacionado |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx\_id\_cli | | X |  |  | id\_cli | | |
| idx\_placa | | X |  |  | placa | | |
| idx\_id\_vaga | | X |  |  | id\_vaga | | |
| idx\_entrada | | X |  |  | entrada | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Reserva | | | | | | |
| **Descrição** | Tabela para armazenar informações sobre reservas | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id | | int | 1 – 2,147,483,647 | NOT NULL | X |  | Identificador da reserva |
| Id\_cli | | varchar | 1 - 2,147,483,647 | NOT NULL |  | X | Referencia o cliente efetuando a reserva |
| Id\_vaga | | int | 1 - 2,147,483,647 | NOT NULL |  | X | Referencia a vaga a ser reservada |
| placa | | varchar | 1 - 20 | NOT NULL |  | X | Referencia a placa do carro que irá ocupar a vaga |
| entrada | | datetime | 1000-01-01 00:00:00 - 9999-12-31 23:59:59 | NOT NULL |  |  | Dia e horário a partir do qual a vaga ficara reservada |
| saida | | datetime | 1000-01-01 00:00:00 - 9999-12-31 23:59:59 | NOT NULL |  |  | Dia e horário a partir do qual a vaga voltara a taxação normal |
| tarifa | | int | 1 - 2,147,483,647 | NOT NULL |  |  | Preço a ser paga pela reserva |
| promoçao | | int | 1 - 2,147,483,647 |  |  | X | Promoção relativa à reserva (não obrigatório) |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
|  | |  |  |  |  | | |
|  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Promoções | | | | | | | |
| **Descrição** | Tabela para armazenar informações sobre promoções/descontos | | | | | | | |
| **Atributos** | | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id | | int | 1 – 2,147,483,647 | NOT NULL | X |  | Identificador da promoção |
| Id\_cli | | int | 1 - 2,147,483,647 | NOT NULL |  | X | Referencia o cliente |
| desconto | | int | 1 - 2,147,483,647 | NOT NULL |  |  | Valor do desconto |
| desck | | varchar | 1 - 255 | NOT NULL |  |  | Descrição da promoção |
| dia | | date | 1000-01-01 – 9999-12-31 | NOT NULL |  |  | Dia em que a promoção será válida |
|  | |  |  |  |  |  |  |
| **Índice** | | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
|  | |  |  |  |  | | |
|  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Fidelidade | | | | | | |
| **Descrição** | Tabela para armazenar informações sobre bônus de fidelidade do cliente | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id | | int | 1 – 2,147,483,647 | NOT NULL | X |  | Identificador do bônus |
| Id\_cli | | int | 1 - 2,147,483,647 | NOT NULL |  | X | Referencia o cliente |
| cota\_minima | | smallint | 1 - 32,767 | NOT NULL |  |  | Mínimo de pontos necessários para o bônus |
| pontos | | int | 1 - 2,147,483,647 | NOT NULL |  |  | Pontos adquiridos pelo cliente |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
|  | |  |  |  |  | | |
|  | |  |  |  |  | | |

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

Create DATABASE estacionamentdb;

CREATE TABLE Cliente (

id INT PRIMARY KEY AUTO\_INCREMENT,

nome VARCHAR(255),

telefone VARCHAR(20),

email VARCHAR(255),

ir VARCHAR(20)

);

CREATE TABLE Veiculo (

placa VARCHAR(20) PRIMARY KEY,

id\_cli INT,

cor VARCHAR(50),

modelo VARCHAR(50),

marca VARCHAR(50),

FOREIGN KEY (id\_cli) REFERENCES Cliente(id)

);

CREATE TABLE Vaga (

id INT PRIMARY KEY AUTO\_INCREMENT,

id\_veiculo VARCHAR(20),

ocupada BOOL,

`desc` VARCHAR(255),

tipo VARCHAR(50),

valor\_hora INT,

FOREIGN KEY (id\_veiculo) REFERENCES Veiculo(placa)

);

CREATE TABLE Funcionario (

id INT PRIMARY KEY AUTO\_INCREMENT,

nome VARCHAR(255),

telefone VARCHAR(20),

email VARCHAR(255),

ir VARCHAR(20),

salario INT

);

CREATE TABLE Assinatura (

id INT PRIMARY KEY AUTO\_INCREMENT,

plano VARCHAR(50),

valor INT,

`desc` VARCHAR(255),

id\_cli INT,

FOREIGN KEY (id\_cli) REFERENCES Cliente(id)

);

CREATE TABLE Pagamentos (

id INT PRIMARY KEY AUTO\_INCREMENT,

id\_cli INT,

id\_vaga INT,

placa VARCHAR(20),

entrada DATETIME,

saida DATETIME,

saldo INT,

total\_horas INT,

FOREIGN KEY (id\_cli) REFERENCES Cliente(id),

FOREIGN KEY (id\_vaga) REFERENCES Vaga(id),

FOREIGN KEY (placa) REFERENCES Veiculo(placa)

);

CREATE TABLE Promocoes (

id INT PRIMARY KEY AUTO\_INCREMENT,

dia DATE,

desconto INT,

`desc` VARCHAR(255)

);

CREATE TABLE Reserva (

id INT PRIMARY KEY AUTO\_INCREMENT,

id\_cli INT,

id\_vaga INT,

placa VARCHAR(20),

entrada DATETIME,

saida DATETIME,

tarifa INT,

promocao INT,

FOREIGN KEY (id\_cli) REFERENCES Cliente(id),

FOREIGN KEY (id\_vaga) REFERENCES Vaga(id),

FOREIGN KEY (placa) REFERENCES Veiculo(placa),

FOREIGN KEY (promocao) REFERENCES Promocoes(id)

);

-- Tabela de fidelidade e recompensas

CREATE TABLE Fidelidade (

id INT PRIMARY KEY AUTO\_INCREMENT,

id\_cli INT,

pontos INT,

FOREIGN KEY (id\_cli) REFERENCES Cliente(id)

);

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

import random

import mysql.connector

from faker import Faker

# Configuração da conexão com o banco de dados

config = {

'user': 'seu\_usuario',

'password': 'sua\_senha',

'host': 'localhost',

'database': 'estacionamento'

}

# Criando a conexão

conn = mysql.connector.connect(\*\*config)

cursor = conn.cursor()

# Inicializando o Faker

fake = Faker()

# Função para inserir dados na tabela Cliente

def insert\_clients(cursor, num\_records):

for \_ in range(num\_records):

nome = fake.name()

telefone = fake.phone\_number()

email = fake.email()

ir = fake.ssn()

cursor.execute("""

INSERT INTO Cliente (nome, telefone, email, ir)

VALUES (%s, %s, %s, %s)

""", (nome, telefone, email, ir))

conn.commit()

# Função para inserir dados na tabela Veiculo

def insert\_vehicles(cursor, num\_records):

cursor.execute("SELECT id FROM Cliente")

client\_ids = [row[0] for row in cursor.fetchall()]

for \_ in range(num\_records):

placa = fake.license\_plate()

id\_cli = random.choice(client\_ids)

cor = fake.color\_name()

modelo = fake.word()

marca = fake.company()

cursor.execute("""

INSERT INTO Veiculo (placa, id\_cli, cor, modelo, marca)

VALUES (%s, %s, %s, %s, %s)

""", (placa, id\_cli, cor, modelo, marca))

conn.commit()

# Função para inserir dados na tabela Vaga

def insert\_vagas(cursor, num\_records):

for \_ in range(num\_records):

id\_veiculo = None

ocupada = fake.boolean()

desc = fake.text(max\_nb\_chars=50)

tipo = random.choice(['Coberta', 'Descoberta'])

valor\_hora = random.randint(5, 20)

cursor.execute("""

INSERT INTO Vaga (id\_veiculo, ocupada, desc, tipo, valor\_hora)

VALUES (%s, %s, %s, %s, %s)

""", (id\_veiculo, ocupada, desc, tipo, valor\_hora))

conn.commit()

# Função para inserir dados na tabela Promocoes

def insert\_promocoes(cursor, num\_records):

for \_ in range(num\_records):

dia = fake.date\_time\_this\_year()

desconto = random.randint(10, 50)

desc = fake.text(max\_nb\_chars=50)

cursor.execute("""

INSERT INTO Promocoes (dia, desconto, desc)

VALUES (%s, %s, %s)

""", (dia, desconto, desc))

conn.commit()

# Função para inserir dados na tabela Reserva

def insert\_reservas(cursor, num\_records):

cursor.execute("SELECT id FROM Cliente")

client\_ids = [row[0] for row in cursor.fetchall()]

cursor.execute("SELECT id FROM Vaga")

vaga\_ids = [row[0] for row in cursor.fetchall()]

for \_ in range(num\_records):

id\_cli = random.choice(client\_ids)

id\_vaga = random.choice(vaga\_ids)

placa = fake.license\_plate()

entrada = fake.date\_time\_this\_year()

saida = fake.date\_time\_this\_year()

tarifa = random.randint(50, 200)

promocao = None

cursor.execute("""

INSERT INTO Reserva (id\_cli, id\_vaga, placa, entrada, saida, tarifa, promocao)

VALUES (%s, %s, %s, %s, %s, %s, %s)

""", (id\_cli, id\_vaga, placa, entrada, saida, tarifa, promocao))

conn.commit()

# Função para inserir dados na tabela Assinatura

def insert\_assinaturas(cursor, num\_records):

cursor.execute("SELECT id FROM Cliente")

client\_ids = [row[0] for row in cursor.fetchall()]

for \_ in range(num\_records):

plano = random.choice(['Mensal', 'Anual'])

valor = random.randint(100, 1000)

desc = fake.text(max\_nb\_chars=50)

id\_cli = random.choice(client\_ids)

cursor.execute("""

INSERT INTO Assinatura (plano, valor, desc, id\_cli)

VALUES (%s, %s, %s, %s)

""", (plano, valor, desc, id\_cli))

conn.commit()

# Função para inserir dados na tabela Fidelidade

def insert\_fidelidades(cursor, num\_records):

cursor.execute("SELECT id FROM Cliente")

client\_ids = [row[0] for row in cursor.fetchall()]

for \_ in range(num\_records):

id\_cli = random.choice(client\_ids)

cota\_minima = random.randint(50, 150)

pontos = random.randint(0, 500)

desconto = fake.boolean()

cursor.execute("""

INSERT INTO Fidelidade (id\_cli, cota\_minima, pontos, desconto)

VALUES (%s, %s, %s, %s)

""", (id\_cli, cota\_minima, pontos, desconto))

conn.commit()

# Função para inserir dados na tabela Funcionario

def insert\_funcionarios(cursor, num\_records):

for \_ in range(num\_records):

nome = fake.name()

telefone = fake.phone\_number()

email = fake.email()

ir = fake.ssn()

salario = random.randint(2000, 5000)

cursor.execute("""

INSERT INTO Funcionario (nome, telefone, email, ir, salario)

VALUES (%s, %s, %s, %s, %s)

""", (nome, telefone, email, ir, salario))

conn.commit()

# Função para inserir dados na tabela Pagamentos

def insert\_pagamentos(cursor, num\_records):

cursor.execute("SELECT id FROM Cliente")

client\_ids = [row[0] for row in cursor.fetchall()]

cursor.execute("SELECT id FROM Funcionario")

funcionario\_ids = [row[0] for row in cursor.fetchall()]

cursor.execute("SELECT id FROM Vaga")

vaga\_ids = [row[0] for row in cursor.fetchall()]

for \_ in range(num\_records):

id\_cli = random.choice(client\_ids)

id\_fun = random.choice(funcionario\_ids)

placa = fake.license\_plate()

id\_vaga = random.choice(vaga\_ids)

entrada = fake.date\_time\_this\_year()

saida = fake.date\_time\_this\_year()

saldo = random.randint(50, 500)

cursor.execute("""

INSERT INTO Pagamentos (id\_cli, id\_fun, placa, id\_vaga, entrada, saida, saldo)

VALUES (%s, %s, %s, %s, %s, %s, %s)

""", (id\_cli, id\_fun, placa, id\_vaga, entrada, saida, saldo))

conn.commit()

# Inserindo 10.000 registros em cada tabela

num\_records = 10000

print("Inserting data into Cliente...")

insert\_clients(cursor, num\_records)

print("Inserting data into Veiculo...")

insert\_vehicles(cursor, num\_records)

print("Inserting data into Vaga...")

insert\_vagas(cursor, num\_records)

print("Inserting data into Promocoes...")

insert\_promocoes(cursor, num\_records)

print("Inserting data into Reserva...")

insert\_reservas(cursor, num\_records)

print("Inserting data into Assinatura...")

insert\_assinaturas(cursor, num\_records)

print("Inserting data into Fidelidade...")

insert\_fidelidades(cursor, num\_records)

print("Inserting data into Funcionario...")

insert\_funcionarios(cursor, num\_records)

print("Inserting data into Pagamentos...")

insert\_pagamentos(cursor, num\_records)

# Fechando a conexão

cursor.close()

conn.close()

print("Data insertion completed.")

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

--1 - Como obter a lista de vagas de estacionamento que não foram ocupadas nos últimos 30 dias?

SELECT v.id, v.`desc`

FROM Vaga v LEFT JOIN Pagamentos p

ON v.id = p.id\_vaga AND p.entrada >= DATE\_SUB(NOW(), INTERVAL 30 DAY)

WHERE p.id IS NULL;

--2 - Qual a taxa de ocupação média de cada vaga de estacionamento por hora nos últimos 7 dias?

SELECT v.`desc` AS descricao,

COALESCE(SUM(TIMESTAMPDIFF(HOUR, p.entrada, p.saida)), 0) / 168 AS taxa\_ocupacao\_media

FROM Vaga v LEFT JOIN Pagamentos p

ON v.id = p.id\_vaga AND p.entrada >= DATE\_SUB(NOW(), INTERVAL 7 DAY)

GROUP BY v.id, v.`desc`;

--3 - Quais as vagas de estacionamento que nunca ficaram disponíveis por mais de 4 horas consecutivas nos últimos 30 dias?

SELECT v.id AS vaga\_id, v.`desc` AS descricao

FROM Vaga v

WHERE tempo\_disponibilidade(v.id) = TRUE;

--4 - Quais os horários de pico de ocupação de vagas de estacionamento nos últimos 30 dias, considerando intervalos de 1 hora?

SELECT DATE\_FORMAT(p.entrada, '%Y-%m-%d %H:00:00') AS hora,

COUNT(\*) AS total\_ocupacoes

FROM Pagamentos p

WHERE p.entrada >= DATE\_SUB(NOW(), INTERVAL 30 DAY)

GROUP BY DATE\_FORMAT(p.entrada, '%Y-%m-%d %H:00:00')

ORDER BY total\_ocupacoes DESC

LIMIT 3;