**DOCUMENTAÇÃO DA ORGANIZAÇÃO E PLANEJAMENTO DO PROJETO “HABITÁE”**

Curso de Administrador de Banco de Dados em MySQL na Firjan Maracanã

Turma 2025

Professor:

*Hamilton*

Alunos:

*Lucas Ribeiro Rodrigues*

*Victhor de Alcantara Félix*

*Vinícius Lima de Oliveira*

| **DAS ATRIBUIÇÕES DO PROJETO:** | | | |
| --- | --- | --- | --- |
| Pessoas Nome | Menus suspensos Função | Menus suspensos Projetos | Menus suspensos Contribuições |
| [Victhor Felix](mailto:victhorfelix7@gmail.com) | Engenharia | Implementação | Prototipagem |
| [lucasrrodrigues2001@gmail.com](mailto:lucasrrodrigues2001@gmail.com) | Engenharia | Pesquisa | Especificações |
| [vinicius.lima9810@gmail.com](mailto:vinicius.lima9810@gmail.com) | Gerenciamento de projetos | Planejamento | Pesquisa |

# **DECISÕES DE MODELAGEM DO BANCO DE DADOS - HABITAE**

## **1. Estratégia de Normalização**

## 1.1 Escolha de Modelagem

O banco de dados foi modelado seguindo a Terceira Forma Normal (3FN) para garantir:

* Eliminação de redundâncias: Dados repetitivos foram extraídos para tabelas especializadas
* Integridade referencial: Relacionamentos bem definidos entre entidades
* Consistência transacional: Operações atômicas mantêm a coerência dos dados

Exemplo prático: A separação das tabelas ENDERECO e IMOVEL evita duplicação de informações geográficas, permitindo que múltiplos imóveis compartilhem o mesmo endereço quando aplicável.

## 1.2 Escolha de Entidades e Atributos

A seleção das 10 entidades principais baseou-se em:

Análise de Domínio:

* Entidades nucleares: IMOVEL, USUARIO, PROPRIETARIO
* Entidades de suporte: ENDERECO, RESERVA, PAGAMENTO
* Entidades corporativas: EMPRESA, FUNCIONARIO
* Entidades documentais: NOTA\_FISCAL, ANUNCIO

Critérios de definição de atributos:

* Atomicidade: Cada atributo contém um único valor
* Relevância: Atributos diretamente relacionados à entidade
* Necessidade: Evitar atributos "por precaução"

## 2. Justificativas para Tipos de Dados

### 2.1 Tipos Textuais

sql

CopyDownload

*-- CNPJ e CPF como VARCHAR para preservar zeros à esquerda*

CNPJ VARCHAR(14) PRIMARY KEY, *-- Formato 12345678000199*

CPF VARCHAR(11) PRIMARY KEY, *-- Formato 12345678901*

*-- Nomes com tamanho adequado para dados brasileiros*

NOME VARCHAR(100) NOT NULL, *-- Compatível com nomes completos*

*-- Email com espaço para endereços longos*

EMAIL VARCHAR(60) NOT NULL, *-- Suporta até 60 caracteres*

Justificativa: O uso de VARCHAR permite flexibilidade no armazenamento enquanto preserva a formatação original dos documentos. Os tamanhos foram definidos com base nas especificações oficiais brasileiras.

### 2.2 Tipos Numéricos e Monetários

sql

CopyDownload

*-- Valores monetários com precisão decimal*

VALOR DECIMAL(10,2) NOT NULL, *-- Até 10 dígitos, 2 casas decimais*

*-- Identificadores numéricos com auto-incremento*

ID\_IMOVEL INT PRIMARY KEY AUTO\_INCREMENT, *-- Faixa suficiente para crescimento*

Justificativa: DECIMAL é ideal para valores monetários por evitar erros de arredondamento. INT para IDs oferece uma faixa de 2+ bilhões de registros, suficiente para o escopo do projeto.

### 2.3 Tipos Especializados

sql

CopyDownload

*-- Enumerações para domínios controlados*

INTENCAO ENUM('Venda', 'Aluguel') NOT NULL,

TIPO ENUM('Apartamento', 'Casa', 'Comercial') NOT NULL,

*-- Datas para controle temporal*

CHECKIN DATE NOT NULL, *-- Apenas data, sem hora*

DATA\_EMISSAO TIMESTAMP DEFAULT CURRENT\_TIMESTAMP *-- Data e hora automática*

Justificativa: ENUM garante consistência nos valores permitidos. DATE para datas simples e TIMESTAMP para registros temporais precisos com fuso horário.

## **3. Fundamentação dos Relacionamentos**

### 3.1 Cardinalidades Implementadas

sql

CopyDownload

*-- Relacionamento 1:N (EMPRESA → IMOVEL)*

ALTER TABLE IMOVEL

ADD CONSTRAINT fk\_imovel\_empresa FOREIGN KEY (CNPJ) REFERENCES EMPRESA(CNPJ)

*-- Relacionamento 1:1 (IMOVEL → ENDERECO)*

ALTER TABLE IMOVEL

ADD CONSTRAINT fk\_imovel\_endereco FOREIGN KEY (ID\_ENDERECO) REFERENCES ENDERECO(ID\_ENDERECO)

*-- Relacionamento N:N (USUARIO → IMOVEL via RESERVA)*

ALTER TABLE RESERVA

ADD CONSTRAINT fk\_reserva\_usuario FOREIGN KEY (CPF\_USUARIO) REFERENCES USUARIO(CPF),

ADD CONSTRAINT fk\_reserva\_imovel FOREIGN KEY (ID\_IMOVEL) REFERENCES IMOVEL(ID\_IMOVEL)

### 3.2 Estratégia de Chaves Estrangeiras

Política de Integridade Referencial:

* ON DELETE RESTRICT: Previne exclusão de registros com dependências
* ON UPDATE CASCADE: Propaga alterações de chaves primárias

Justificativa: Esta abordagem previne dados órfãos enquanto mantém a consistência dos relacionamentos durante operações de atualização.

## **4. Considerações de Performance e Escalabilidade**

### 4.1 Estratégia de Indexação

Índices Implementados:

* Chaves primárias: Indexação automática por otimização do MySQL
* Chaves estrangeiras: Indexação para acelerar joins
* Campos de busca frequente: IMOVEL.TIPO, IMOVEL.INTENCAO

### 4.2 Otimização de Espaço

Técnicas aplicadas:

* Tipos de dados adequados ao conteúdo esperado
* Uso de NULL apenas quando semanticamente apropriado
* Normalização para minimizar redundância

## **5. Decisões de Design para Flexibilidade Futura**

### 5.1 Extensibilidade do Modelo

Características implementadas:

* Tabelas genéricas para endereços e contatos
* Enumerações expansíveis para novos tipos
* Estrutura modular para adição de novas entidades

### 5.2 Manutenibilidade

Práticas adotadas:

* Nomenclatura consistente e autoexplicativa
* Documentação embutida via comentários SQL
* Estrutura modular para fácil evolução

## **6. Conformidade com Regulamentações**

### 6.1 LGPD (Lei Geral de Proteção de Dados)

Medidas implementadas:

* Separação de dados pessoais em tabelas específicas
* Controle de acesso através de views e procedures
* Capacidade de anonimização para dados sensíveis

### 6.2 Normas Contábeis Brasileiras

Adequações:

* Campos para CNPJ/CPF formatados conforme padrões nacionais
* Suporte a notas fiscais eletrônicas
* Registro temporal completo para auditoria

Resumo Técnico: O modelo HABITAE foi arquitetado seguindo as melhores práticas de engenharia de dados, balanceando performance, integridade e flexibilidade. As decisões técnicas foram fundamentadas em requisitos de negócio, normas regulatórias e projeções de crescimento futuro, resultando em uma base sólida para o sistema imobiliário.

Esta documentação serve como guia de referência para desenvolvedores, administradores de banco de dados e equipe de negócios, garantindo compreensão consistente da estrutura implementada.

# **DOCUMENTAÇÃO DAS CONSULTAS SQL - HABITAE**

## **1. Consultas com SELECT e WHERE**

### 1.1 Listar todos os imóveis para venda

*-- Listar todos os imóveis para venda*

SELECT \* FROM IMOVEL WHERE INTENCAO = 'Venda';

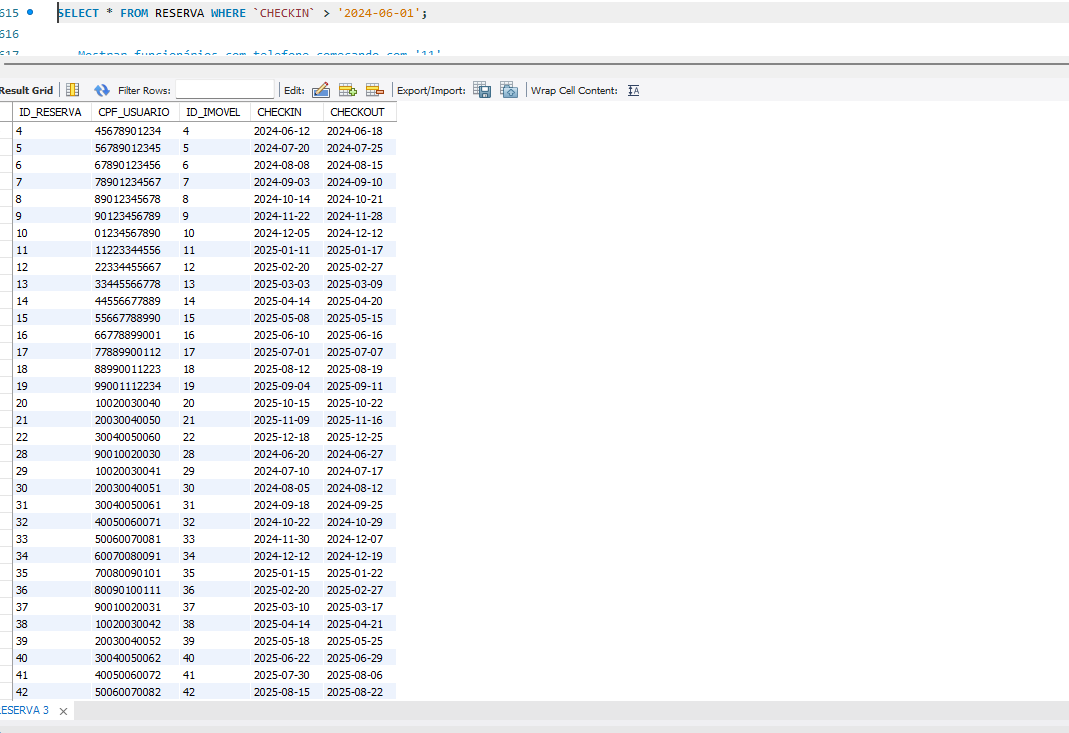
### 

Explicação: Esta consulta retorna todos os registros da tabela IMOVEL onde a intenção é especificamente "Venda", permitindo identificar quais propriedades estão disponíveis para compra.

### 1.2 Selecionar reservas com check-in após 01/06/2024

*-- Selecionar reservas com check-in após 01/06/2024*

SELECT \* FROM RESERVA WHERE `CHECKIN` > '2024-06-01';

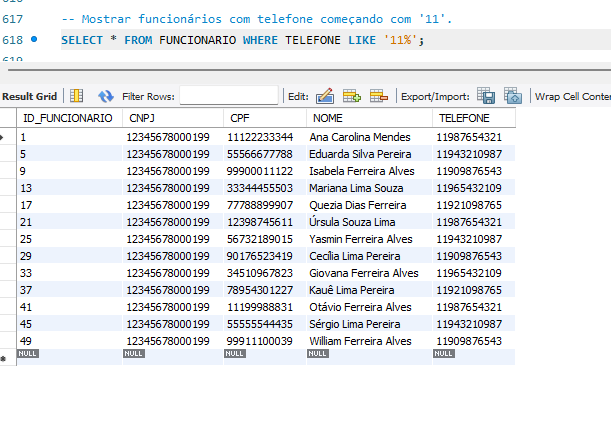


Explicação: Esta consulta filtra as reservas que têm data de check-in posterior a 1º de junho de 2024, útil para analisar reservas futuras.

### 1.3 Mostrar funcionários com telefone começando com '11'

*-- Mostrar funcionários com telefone começando com '11'*

SELECT \* FROM FUNCIONARIO WHERE TELEFONE LIKE '11%';

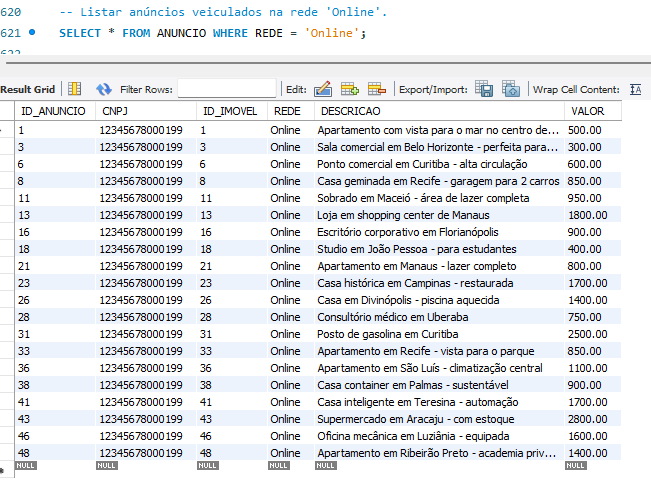


Explicação: Localiza funcionários cujos números de telefone começam com '11' (código de área de São Paulo), permitindo identificar colaboradores baseados nessa região.

### 1.4 Listar anúncios veiculados na rede 'Online'

*-- Listar anúncios veiculados na rede 'Online'*

SELECT \* FROM ANUNCIO WHERE REDE = 'Online';

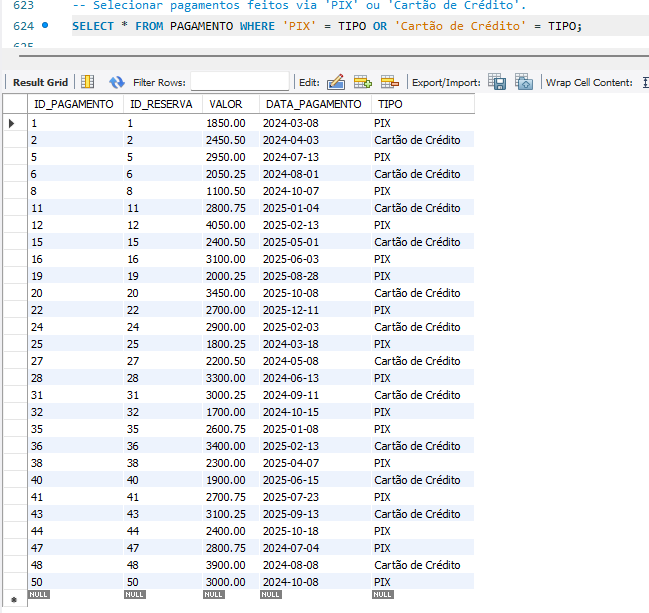


Explicação: Retorna todos os anúncios que foram veiculados através da rede 'Online', facilitando a análise do marketing digital da empresa.

### 1.5 Selecionar pagamentos feitos via 'PIX' ou 'Cartão de Crédito'

*-- Selecionar pagamentos feitos via 'PIX' ou 'Cartão de Crédito'*

SELECT \* FROM PAGAMENTO WHERE TIPO = 'PIX' OR TIPO = 'Cartão de Crédito';



Explicação: Identifica todos os pagamentos realizados através dos métodos PIX ou Cartão de Crédito, auxiliando na análise das preferências de pagamento dos clientes.

## **2. Consultas com GROUP BY e ORDER BY**

### 2.1 Contar o número de imóveis por tipo

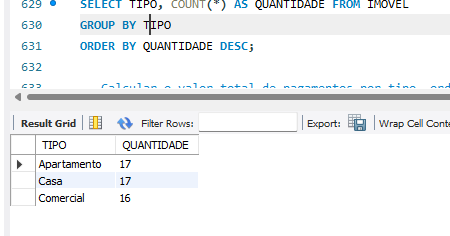
*-- Contar o número de imóveis por tipo, ordenado pela quantidade*

SELECT TIPO, COUNT(\*) AS QUANTIDADE

FROM IMOVEL

GROUP BY TIPO

ORDER BY QUANTIDADE DESC;



Explicação: Agrupa os imóveis por tipo (Apartamento, Casa, Comercial) e conta quantos existem de cada, ordenando do mais comum para o menos comum.

### 2.2 Calcular o valor total de pagamentos por tipo

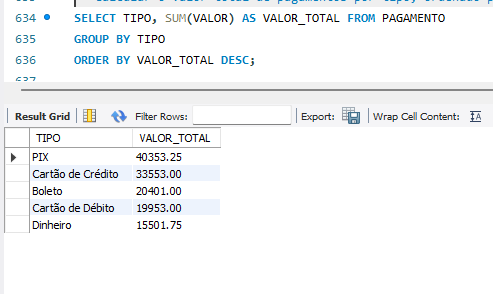
*-- Calcular o valor total de pagamentos por tipo, ordenado pelo valor*

SELECT TIPO, SUM(VALOR) AS VALOR\_TOTAL

FROM PAGAMENTO

GROUP BY TIPO

ORDER BY VALOR\_TOTAL DESC;



Explicação: Calcula o valor total transacionado para cada método de pagamento, ordenando do maior para o menor volume financeiro.

### 2.3 Média de valores de anúncios por rede de divulgação

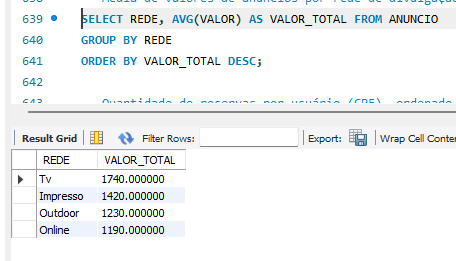
*-- Média de valores de anúncios por rede de divulgação*

SELECT REDE, AVG(VALOR) AS MEDIA\_VALOR

FROM ANUNCIO

GROUP BY REDE

ORDER BY MEDIA\_VALOR DESC;



Explicação: Calcula o valor médio investido em anúncios por tipo de rede de divulgação, ordenando das redes mais caras para as mais econômicas.

### 

### 

### 2.4 Quantidade de reservas por usuário

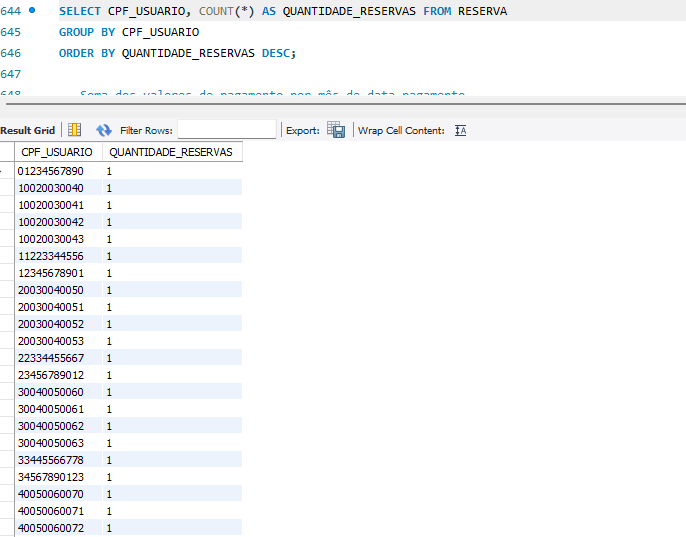
*-- Quantidade de reservas por usuário (CPF), ordenado por quantidade*

SELECT CPF\_USUARIO, COUNT(\*) AS QUANTIDADE\_RESERVAS

FROM RESERVA

GROUP BY CPF\_USUARIO

ORDER BY QUANTIDADE\_RESERVAS DESC;



Explicação: Conta quantas reservas cada usuário realizou, ordenando dos usuários mais frequentes para os menos frequentes.

### 2.5 Soma dos valores de pagamento por mês

*-- Soma dos valores de pagamento por mês de data\_pagamento*

SELECT

MONTH(DATA\_PAGAMENTO) AS MES,

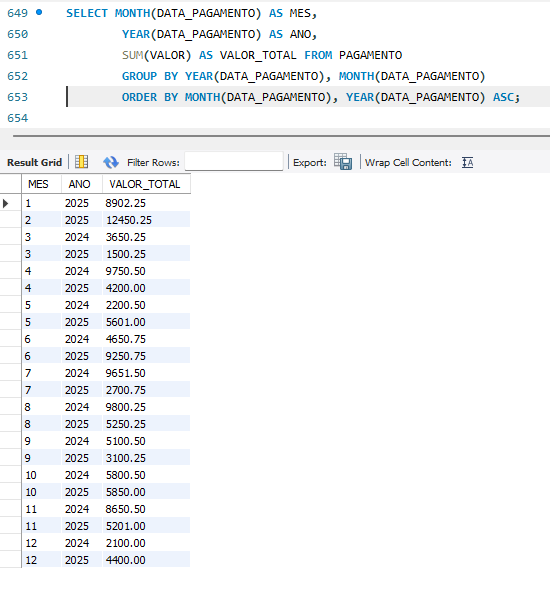
YEAR(DATA\_PAGAMENTO) AS ANO,

SUM(VALOR) AS VALOR\_TOTAL

FROM PAGAMENTO

GROUP BY YEAR(DATA\_PAGAMENTO), MONTH(DATA\_PAGAMENTO)

ORDER BY ANO, MES;



Explicação: Agrupa os pagamentos por mês e ano, calculando o total recebido em cada período, ordenado cronologicamente.

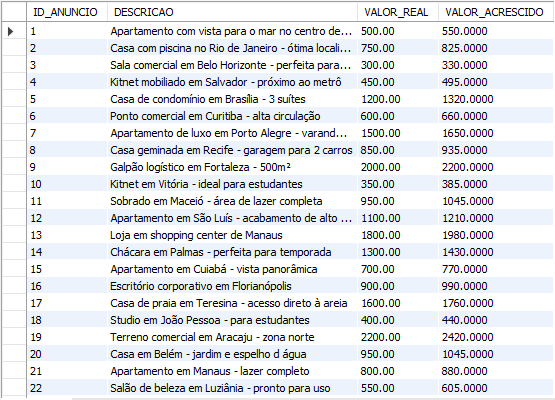
## **3. Consultas com Operadores Aritméticos**

### 3.1 Calcular o valor do anúncio com acréscimo de 10%

sql

SELECT ID\_ANUNCIO, DESCRICAO, VALOR AS VALOR\_REAL,

VALOR \* 1.10 AS VALOR\_ACRESCIDO FROM ANUNCIO;



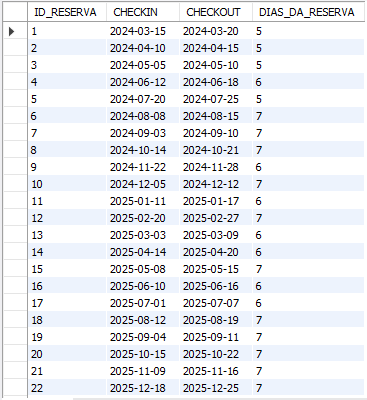
Explicação: Mostra o valor original dos anúncios e simula um aumento de 10%, útil para projeções de reajuste de preços.

### 3.2 Diferença em dias entre check-in e check-out

sql

SELECT ID\_RESERVA, CHECKIN, CHECKOUT,

DATEDIFF(CHECKOUT, CHECKIN) AS DIAS\_DA\_RESERVA FROM RESERVA;



Explicação: Calcula a duração de cada reserva em dias, informação importante para cálculo de valores e estatísticas de permanência.

### 3.3 Valor médio diário das reservas

sql

SELECT R.ID\_RESERVA, R.CHECKIN, R.CHECKOUT, P.VALOR AS VALOR\_TOTAL,

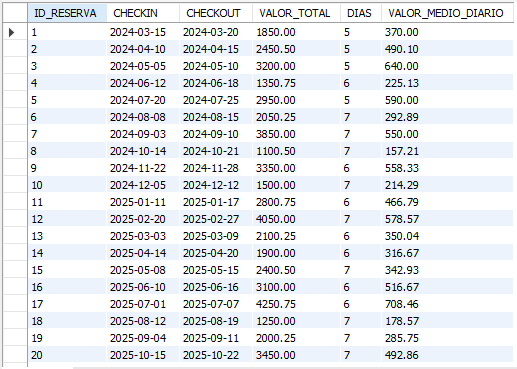
DATEDIFF(R.CHECKOUT, R.CHECKIN) AS DIAS,

ROUND(P.VALOR / DATEDIFF(R.CHECKOUT, R.CHECKIN), 2) AS VALOR\_MEDIO\_DIARIO

FROM RESERVA R

JOIN PAGAMENTO P ON R.ID\_RESERVA = P.ID\_RESERVA

ORDER BY R.ID\_RESERVA;



Explicação: Calcula o valor médio por dia de cada reserva, permitindo análise da taxa diária praticada.

### 3.4 Desconto de 5% nos valores de pagamento acima de R$ 3000

sql

SELECT ID\_PAGAMENTO, VALOR AS VALOR\_REAL,

VALOR \* 0.05 AS VALOR\_DESCONTO,

VALOR - (VALOR \* 0.05) AS VALOR\_TOTAL

FROM PAGAMENTO

WHERE VALOR > 3000;



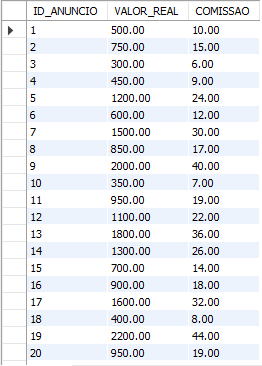
Explicação: Simula um desconto de 5% para pagamentos acima de R$ 3000, mostrando o valor original, o desconto e o novo valor.

### 3.5 Calcular 2% de comissão sobre o valor dos anúncios

sql

SELECT ID\_ANUNCIO, VALOR AS VALOR\_REAL,

ROUND((VALOR \* 0.02), 2) AS COMISSAO FROM ANUNCIO;



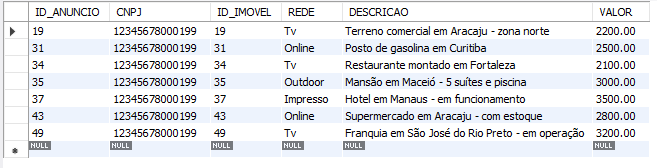
Explicação: Calcula uma comissão de 2% sobre o valor de cada anúncio, útil para cálculo de remuneração de corretores.

## **4. Consultas com Operadores de Comparação**

### 4.1 Selecionar imóveis com valor de anúncio superior a R$ 2000

sql

SELECT \* FROM ANUNCIO WHERE VALOR > 2000;



Explicação: Identifica anúncios de alto valor, acima de R$ 2000, para análise de imóveis premium.

### 4.2 Reservas com mais de 6 dias de duração

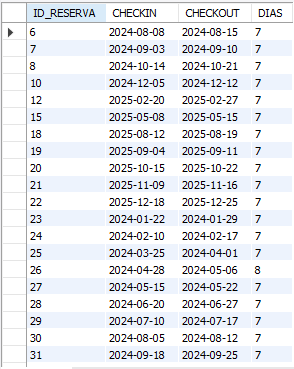
sql

SELECT ID\_RESERVA, CHECKIN, CHECKOUT,

DATEDIFF(CHECKOUT, CHECKIN) AS DIAS

FROM RESERVA

WHERE DATEDIiff(CHECKOUT, CHECKIN) > 6;



Explicação: Filtra reservas com estadias longas (mais de 6 dias), úteis para identificar padrões de hospedagem extendida.

### 

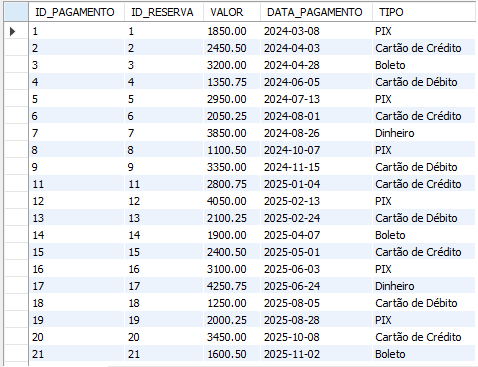
### 

### 

### 4.3 Pagamentos com valor diferente de R$ 1500

sql

SELECT \* FROM PAGAMENTO WHERE VALOR != 1500;



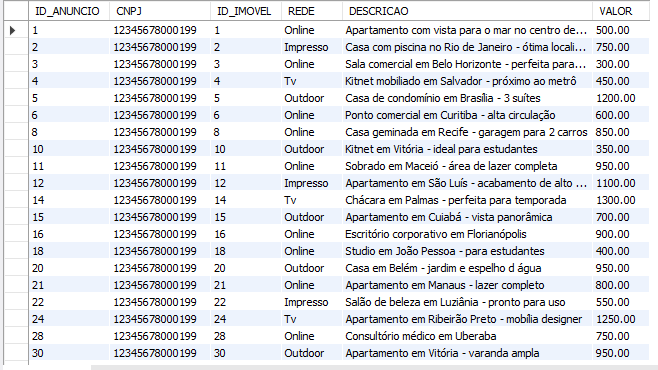
Explicação: Encontra todos os pagamentos que não são exatamente R$ 1500, útil para identificar transações com valores incomuns.

### 4.4 Anúncios com valor menor que a média geral

sql

SELECT \* FROM ANUNCIO

WHERE VALOR < (SELECT AVG(VALOR) FROM ANUNCIO);



Explicação: Seleciona anúncios com valor abaixo da média, possivelmente identificando oportunidades ou imóveis com preços mais acessíveis.

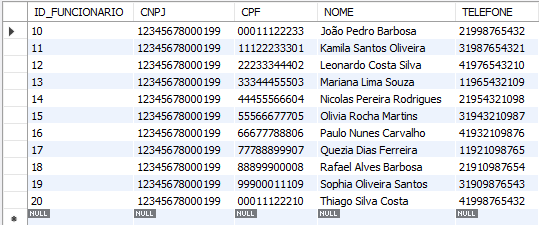
### 4.5 Funcionários com ID entre 10 e 20

sql

SELECT \* FROM FUNCIONARIO

WHERE ID\_FUNCIONARIO BETWEEN 10 AND 20

ORDER BY ID\_FUNCIONARIO;



Explicação: Recupera informações de funcionários com IDs num intervalo específico (10 a 20), útil para análise segmentada.

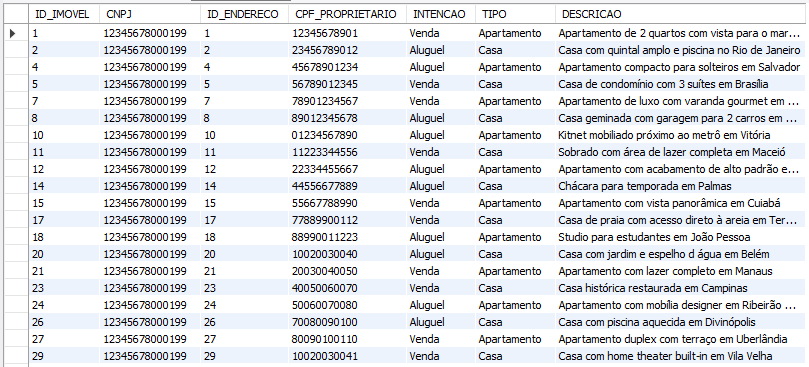
## **5. Consultas com Operadores Lógicos**

### 5.1 Imóveis para aluguel do tipo 'Apartamento' ou 'Casa'

sql

SELECT \* FROM IMOVEL

WHERE INTENCAO = 'Aluguel' AND (TIPO = 'Apartamento' OR TIPO = 'Casa');



Explicação: Encontra imóveis disponíveis para aluguel que são exclusivamente residenciais (apartamentos ou casas).

### 5.2 Reservas em 2024 com valor superior a R$ 2000 ou duração > 5 dias

sql

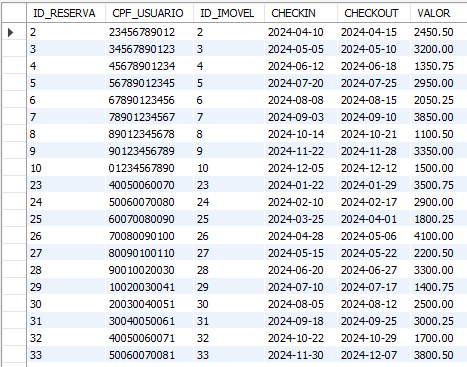
SELECT R.\*, P.VALOR

FROM RESERVA R

JOIN PAGAMENTO P ON R.ID\_RESERVA = P.ID\_RESERVA

WHERE YEAR(R.CHECKIN) = 2024

AND (P.VALOR > 2000 OR DATEDIFF(R.CHECKOUT, R.CHECKIN) > 5);



Explicação: Identifica reservas em 2024 que são either de alto valor (acima de R$ 2000) ou longa duração (mais de 5 dias).

### 5.3 Usuários com telefone da região 11 (SP) ou 21 (RJ)

sql

SELECT \* FROM USUARIO

WHERE TELEFONE LIKE '11%' OR TELEFONE LIKE '21%';



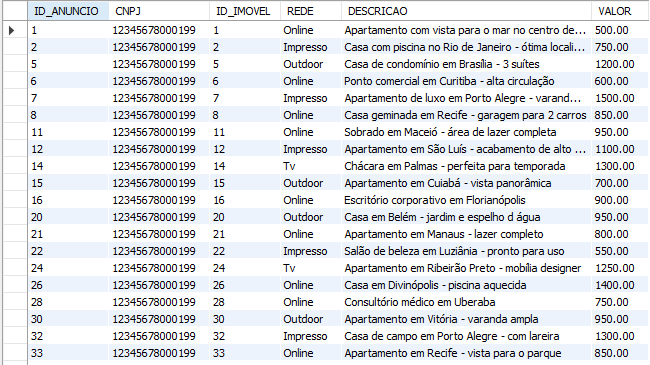
Explicação: Localiza usuários com telefones das regiões de São Paulo (11) ou Rio de Janeiro (21), permitindo análise geográfica.

### 5.4 Anúncios online com valor entre R$ 500 e R$ 1500

sql

SELECT \* FROM ANUNCIO

WHERE REDE = 'Online' AND VALOR BETWEEN 500 AND 1500;



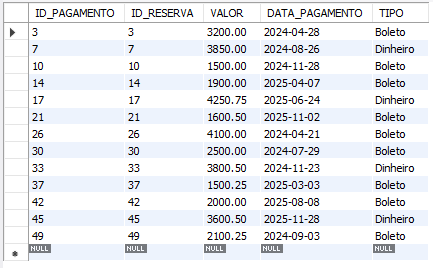
Explicação: Encontra anúncios online com valores em uma faixa específica, útil para análise de estratégia de preços.

### 5.5 Pagamentos via 'Boleto' ou 'Dinheiro' com valor > R$ 1000

sql

SELECT \* FROM PAGAMENTO

WHERE (TIPO = 'Boleto' OR TIPO = 'Dinheiro') AND VALOR > 1000;



Explicação: Identifica pagamentos de alto valor realizados em espécie (dinheiro) ou boleto, importantes para controle financeiro.

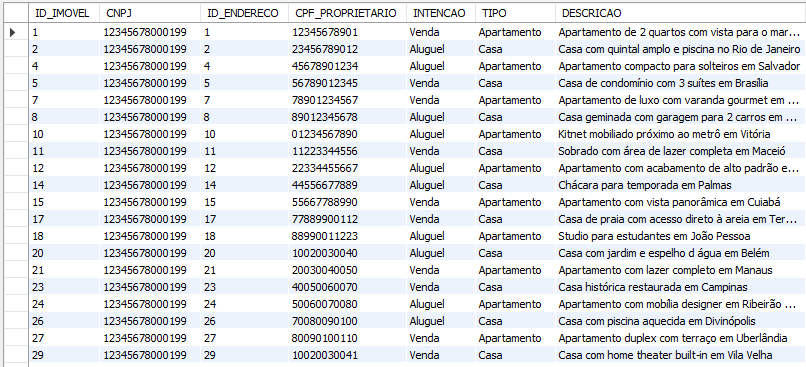
## **6. Consultas com Operadores Lógicos e Negação**

### 6.1 Imóveis que não são do tipo 'Comercial'

sql

SELECT \* FROM IMOVEL

WHERE NOT TIPO = 'Comercial';



Explicação: Retorna todos os imóveis que não são comerciais, ou seja, apenas residenciais.

### 6.2 Reservas que não foram pagas com 'Cartão de Crédito'

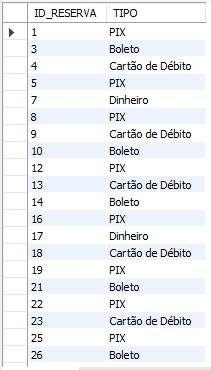
sql

SELECT R.ID\_RESERVA, P.TIPO

FROM RESERVA R

JOIN PAGAMENTO P ON R.ID\_RESERVA = P.ID\_RESERVA

WHERE NOT P.TIPO = 'Cartão de Crédito';



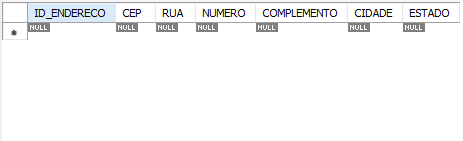
Explicação: Identifica reservas pagas com métodos alternativos ao cartão de crédito, útil para análise de preferências de pagamento.

## **7. Consultas com Operadores Auxiliares**

### 7.1 Endereços com complemento nulo

sql

SELECT \* FROM ENDERECO WHERE COMPLEMENTO IS NULL;



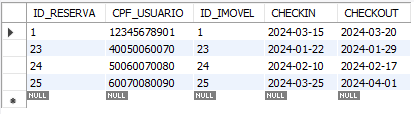
Explicação: Encontra endereços que não possuem complemento, possivelmente necessitando de complementação de dados. Como demonstrado na imagem, não há complementos vazios.

### 7.2 Reservas com check-in entre Janeiro e Março de 2024

sql

SELECT \* FROM RESERVA

WHERE CHECKIN BETWEEN '2024-01-01' AND '2024-03-31';

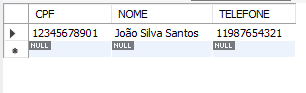


Explicação: Seleciona reservas do primeiro trimestre de 2024, útil para análise sazonal.

### 7.3 Proprietários com 'Silva' no nome

sql

SELECT \* FROM PROPRIETARIO WHERE NOME LIKE '%Silva%';



Explicação: Localiza proprietários com "Silva" no nome, demonstrando uso de padrões textuais em consultas.

### 7.4 Imóveis localizados em SP ou RJ

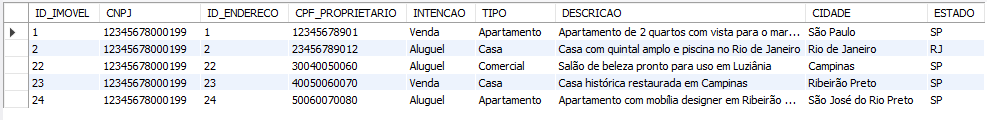
sql

SELECT I.\*, E.CIDADE, E.ESTADO

FROM IMOVEL I

JOIN ENDERECO E ON I.ID\_ENDERECO = E.ID\_ENDERECO

WHERE E.ESTADO IN ('SP', 'RJ');



Explicação: Identifica imóveis situados nos estados de São Paulo ou Rio de Janeiro, usando operador IN.

### 7.5 Usuários com email do domínio '@[email.com](https://email.com/)'

sql

SELECT \* FROM USUARIO WHERE EMAIL LIKE '%@email.com';



Explicação: Encontra usuários com emails de um domínio específico, útil para segmentação de comunicação.

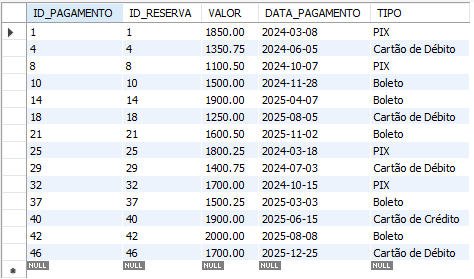
### 

### 

### 7.6 Pagamentos com valores entre R$ 1000 e R$ 2000

sql

SELECT \* FROM PAGAMENTO WHERE VALOR BETWEEN 1000 AND 2000;

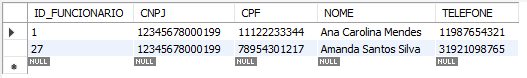


Explicação: Seleciona pagamentos em uma faixa de valor específica, para análise financeira.

### 7.7 Funcionários com nomes começando com 'A'

sql

SELECT \* FROM FUNCIONARIO WHERE NOME LIKE 'A%';



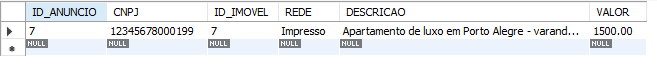
Explicação: Lista funcionários cujos nomes começam com a letra 'A', útil para listagens alfabéticas.

### 

### 7.8 Anúncios com descrição contendo 'luxo'

sql

SELECT \* FROM ANUNCIO WHERE DESCRICAO LIKE '%luxo%';



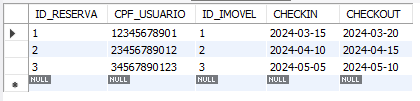
Explicação: Encontra anúncios que mencionam "luxo" na descrição, identificando propriedades premium.

### 7.9 Reservas feitas por usuários específicos

sql

SELECT \* FROM RESERVA

WHERE CPF\_USUARIO IN ('12345678901', '23456789012', '34567890123');



Explicação: Seleciona reservas de um conjunto específico de usuários, usando operador IN.

### 

### 

### 

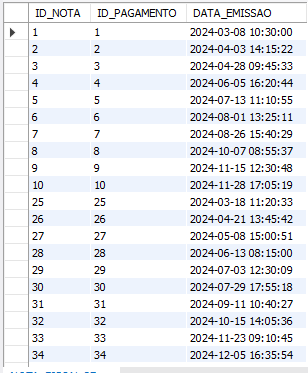
### 

### 7.10 Notas fiscais emitidas em 2024

sql

SELECT \* FROM NOTA\_FISCAL

WHERE DATA\_EMISSAO BETWEEN '2024-01-01 00:00:00' AND '2024-12-31 23:59:59';



Explicação: Recupera todas as notas fiscais emitidas no ano de 2024, para fins contábeis e fiscais.

## **8. Consultas com Funções de Agregação**

### 8.1 Soma total investida em anúncios

sql

SELECT SUM(VALOR) AS TOTAL\_INVESTIDO FROM ANUNCIO;



Explicação: Calcula o investimento total em campanhas publicitárias/anúncios.

### 8.2 Valor médio das reservas

sql

SELECT AVG(P.VALOR) AS MEDIA\_VALOR\_RESERVAS

FROM RESERVA R

JOIN PAGAMENTO P ON R.ID\_RESERVA = P.ID\_RESERVA;



Explicação: Calcula o ticket médio das reservas realizadas.

### 8.3 Maior valor de pagamento registrado

sql

SELECT MAX(VALOR) AS MAIOR\_PAGAMENTO FROM PAGAMENTO;



Explicação: Identifica o maior valor já pago em uma reserva.

### 8.4 Número total de imóveis por cidade

sql

SELECT E.CIDADE, COUNT(I.ID\_IMOVEL) AS TOTAL\_IMOVEIS

FROM IMOVEL I

JOIN ENDERECO E ON I.ID\_ENDERECO = E.ID\_ENDERECO

GROUP BY E.CIDADE

ORDER BY TOTAL\_IMOVEIS DESC;



Explicação: Conta quantos imóveis existem em cada cidade, ordenando das cidades com mais imóveis para as com menos.

### 8.5 Média de valor dos anúncios por tipo de imóvel

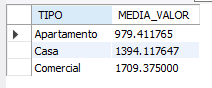
sql

SELECT I.TIPO, AVG(A.VALOR) AS MEDIA\_VALOR

FROM ANUNCIO A

JOIN IMOVEL I ON A.ID\_IMOVEL = I.ID\_IMOVEL

GROUP BY I.TIPO;



Explicação: Calcula o valor médio dos anúncios para cada tipo de imóvel (Apartamento, Casa, Comercial).

## 

## **9. Consultas com Funções de Data**

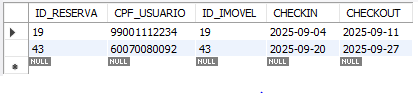
### 9.1 Reservas do mês atual

sql

SELECT \* FROM RESERVA

WHERE MONTH(CHECKIN) = MONTH(CURRENT\_DATE())

AND YEAR(CHECKIN) = YEAR(CURRENT\_DATE());



Explicação: Identifica reservas com check-in no mês corrente, útil para análise de ocupação atual.

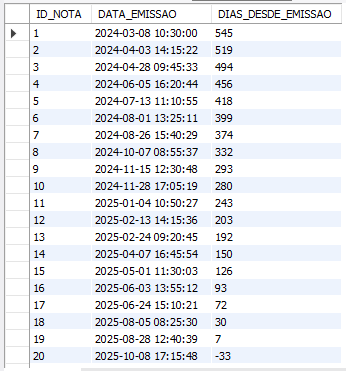
### 9.2 Idade das notas fiscais em dias

sql

SELECT ID\_NOTA, DATA\_EMISSAO,

TIMESTAMPDIFF(DAY, DATA\_EMISSAO, NOW()) AS DIAS\_DESDE\_EMISSAO

FROM NOTA\_FISCAL;



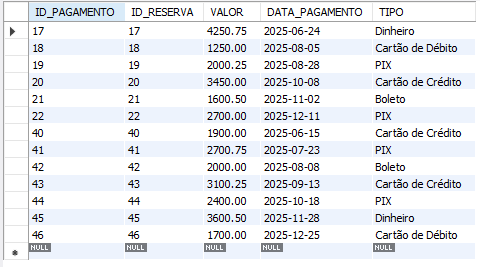
Explicação: Calcula há quantos dias cada nota fiscal foi emitida.

### 9.3 Pagamentos realizados no último trimestre

sql

SELECT \* FROM PAGAMENTO

WHERE DATA\_PAGAMENTO >= DATE\_SUB(CURRENT\_DATE(), INTERVAL 3 MONTH);



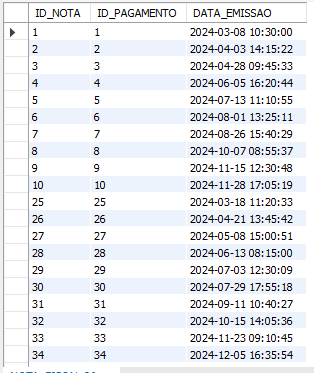
Explicação: Recupera pagamentos dos últimos 3 meses, útil para análise financeira recente.

### 9.4 Notas fiscais emitidas no ano anterior

sql

SELECT \* FROM NOTA\_FISCAL

WHERE YEAR(DATA\_EMISSAO) = YEAR(CURRENT\_DATE()) - 1;



Explicação: Seleciona notas fiscais do ano anterior ao atual, para comparações anuais.

### 9.5 Dias até o check-in das reservas futuras

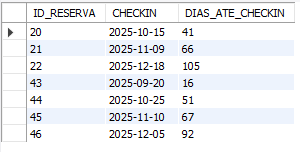
sql

SELECT ID\_RESERVA, CHECKIN,

DATEDIFF(CHECKIN, CURRENT\_DATE()) AS DIAS\_ATE\_CHECKIN

FROM RESERVA

WHERE CHECKIN > CURRENT\_DATE();



Explicação: Calcula quantos dias faltam para o check-in de reservas futuras.

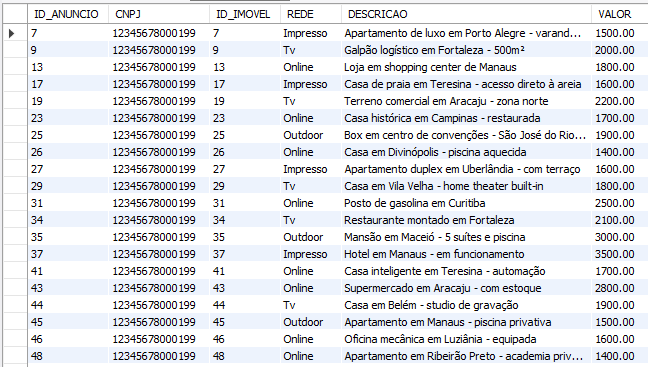
## **10. Subconsultas com Agrupamento**

### 10.1 Imóveis com valor de anúncio acima da média

sql

SELECT \* FROM ANUNCIO

WHERE VALOR > (SELECT AVG(VALOR) FROM ANUNCIO);



Explicação: Identifica anúncios com valor acima da média do mercado.

### 

### 10.2 Usuários com mais reservas que a média

sql

SELECT U.\*, COUNT(R.ID\_RESERVA) AS TOTAL\_RESERVAS

FROM USUARIO U

JOIN RESERVA R ON U.CPF = R.CPF\_USUARIO

GROUP BY U.CPF

HAVING COUNT(R.ID\_RESERVA) > (SELECT AVG(reserva\_count)

FROM (SELECT COUNT(\*) AS reserva\_count

FROM RESERVA

GROUP BY CPF\_USUARIO) AS counts);



Explicação: Encontra usuários com número de reservas acima da média. Mas como os usuários foram registrados com a mesma quantidade de reservas, a consulta não trás nada.

### 10.3 Proprietários com mais de 2 imóveis

sql

SELECT P.\*, COUNT(I.ID\_IMOVEL) AS TOTAL\_IMOVEIS

FROM PROPRIETARIO P

JOIN IMOVEL I ON P.CPF = I.CPF\_PROPRIETARIO

GROUP BY P.CPF

HAVING COUNT(I.ID\_IMOVEL) > 2;



Explicação: Identifica proprietários com portfólio de mais de 2 imóveis. Como cada proprietário tem apenas 1 imovel, a consulta não trás nada.

### 

### 

### 10.4 Redes com investimento total superior a R$ 5000

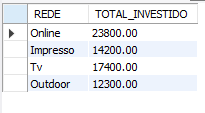
sql

SELECT REDE, SUM(VALOR) AS TOTAL\_INVESTIDO

FROM ANUNCIO

GROUP BY REDE

HAVING SUM(VALOR) > 5000;



Explicação: Mostra redes de publicidade onde o investimento total ultrapassou R$ 5000.

### 10.5 Reservas com valor acima da média por tipo de imóvel

sql

SELECT R.\*, P.VALOR, I.TIPO

FROM RESERVA R

JOIN PAGAMENTO P ON R.ID\_RESERVA = P.ID\_RESERVA

JOIN IMOVEL I ON R.ID\_IMOVEL = I.ID\_IMOVEL

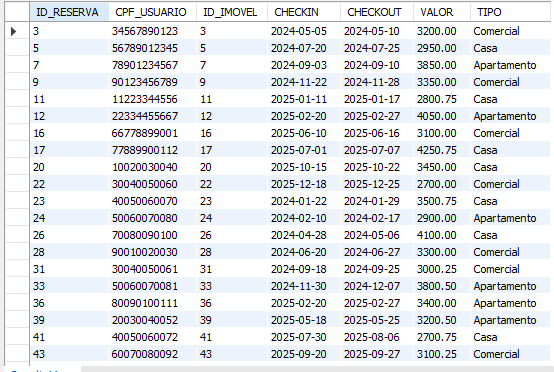
WHERE P.VALOR > (SELECT AVG(P2.VALOR)

FROM PAGAMENTO P2

JOIN RESERVA R2 ON P2.ID\_RESERVA = R2.ID\_RESERVA

JOIN IMOVEL I2 ON R2.ID\_IMOVEL = I2.ID\_IMOVEL

WHERE I2.TIPO = I.TIPO);



Explicação: Identifica reservas com valor acima da média para seu tipo de imóvel.

## **11. Consultas com JOIN**

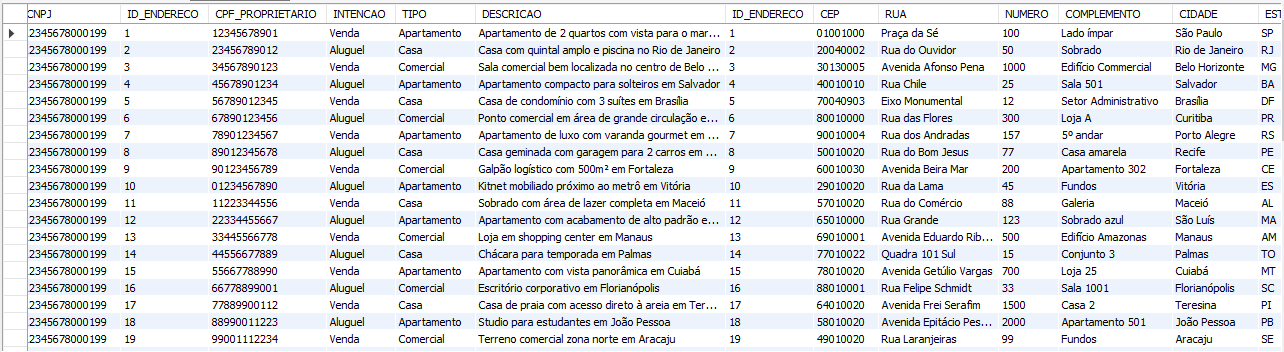
### 11.1 Imóveis com endereços completos

sql

SELECT I.\*, E.\*

FROM IMOVEL I

JOIN ENDERECO E ON I.ID\_ENDERECO = E.ID\_ENDERECO;



Explicação: Combina informações de imóveis com seus respectivos endereços.

### 11.2 Reservas com dados do usuário e imóvel

sql

SELECT R.\*, U.NOME AS NOME\_USUARIO, I.\*

FROM RESERVA R

JOIN USUARIO U ON R.CPF\_USUARIO = U.CPF

JOIN IMOVEL I ON R.ID\_IMOVEL = I.ID\_IMOVEL;



Explicação: Mostra informações completas das reservas incluindo dados do usuário e do imóvel.

### 11.3 Pagamentos com informações da reserva e usuário

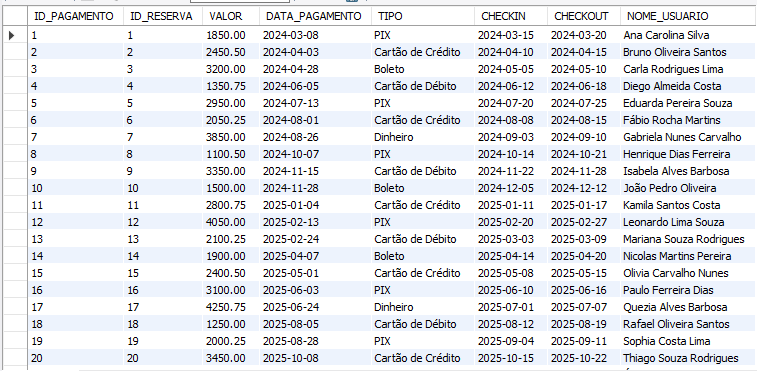
sql

SELECT P.\*, R.CHECKIN, R.CHECKOUT, U.NOME AS NOME\_USUARIO

FROM PAGAMENTO P

JOIN RESERVA R ON P.ID\_RESERVA = R.ID\_RESERVA

JOIN USUARIO U ON R.CPF\_USUARIO = U.CPF;



Explicação: Relaciona pagamentos com informações das reservas e usuários correspondentes.

### 11.4 Anúncios com dados do imóvel e proprietário

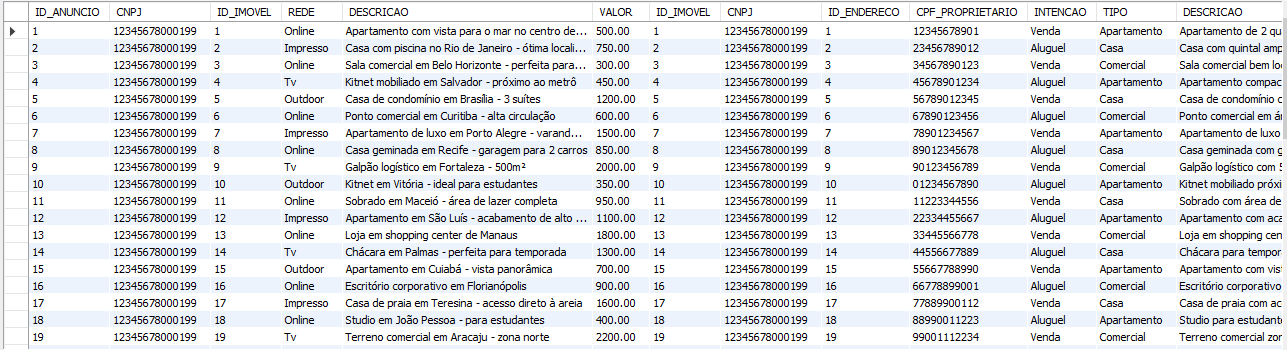
sql

SELECT A.\*, I.\*, P.NOME AS NOME\_PROPRIETARIO

FROM ANUNCIO A

JOIN IMOVEL I ON A.ID\_IMOVEL = I.ID\_IMOVEL

JOIN PROPRIETARIO P ON I.CPF\_PROPRIETARIO = P.CPF;



Explicação: Combina informações de anúncios com dados do imóvel e proprietário.

### 

### 

### 11.5 Notas fiscais com detalhes do pagamento e reserva

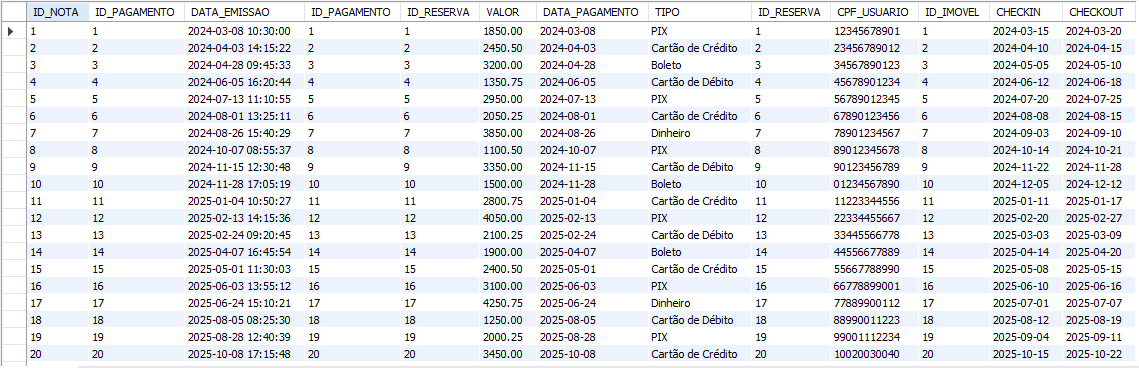
sql

SELECT NF.\*, P.\*, R.\*

FROM NOTA\_FISCAL NF

JOIN PAGAMENTO P ON NF.ID\_PAGAMENTO = P.ID\_PAGAMENTO

JOIN RESERVA R ON P.ID\_RESERVA = R.ID\_RESERVA;



Explicação: Relaciona notas fiscais com informações de pagamento e reserva.

## **12. Consultas com Diferentes Tipos de JOIN**

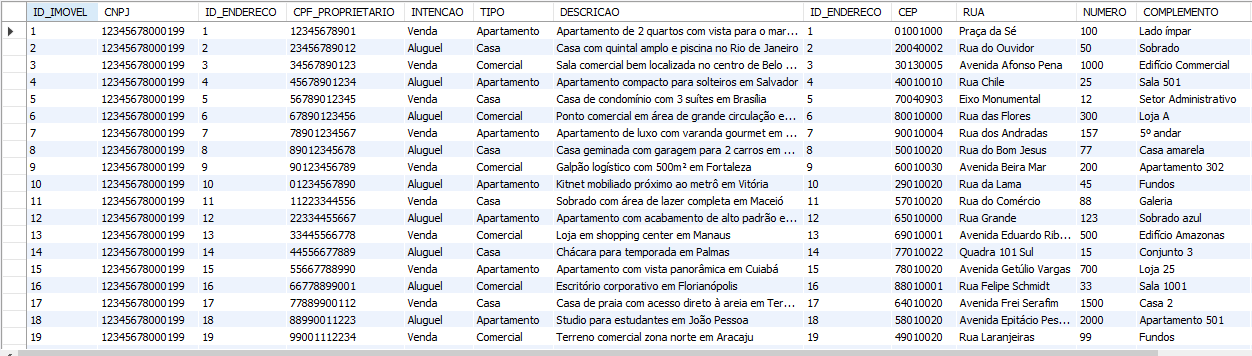
### 12.1 INNER JOIN: Imóveis com endereços válidos

sql

SELECT I.\*, E.\*

FROM IMOVEL I

INNER JOIN ENDERECO E ON I.ID\_ENDERECO = E.ID\_ENDERECO;



Explicação: Retorna apenas imóveis que possuem endereço cadastrado.

### 12.2 LEFT JOIN: Todos os proprietários, mesmo sem imóveis

sql

SELECT P.\*, I.ID\_IMOVEL

FROM PROPRIETARIO P

LEFT JOIN IMOVEL I ON P.CPF = I.CPF\_PROPRIETARIO;



Explicação: Lista todos os proprietários, mostrando NULL para os que não têm imóveis cadastrados.

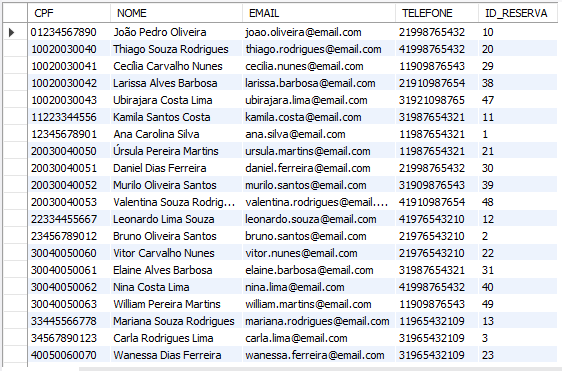
### 12.3 RIGHT JOIN: Todos os usuários, mesmo sem reservas

sql

SELECT U.\*, R.ID\_RESERVA

FROM RESERVA R

RIGHT JOIN USUARIO U ON R.CPF\_USUARIO = U.CPF;



Explicação: Lista todos os usuários, mostrando NULL para os que não fizeram reservas.

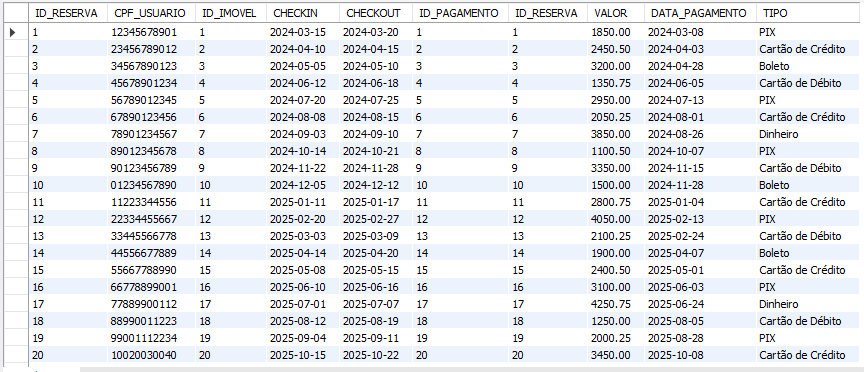
### 12.4 INNER JOIN: Reservas com pagamentos realizados

sql

SELECT R.\*, P.\*

FROM RESERVA R

INNER JOIN PAGAMENTO P ON R.ID\_RESERVA = P.ID\_RESERVA;



Explicação: Retorna apenas reservas que possuem pagamento associado.

### 12.5 LEFT JOIN: Imóveis mesmo sem anúncios

sql

SELECT I.\*, A.ID\_ANUNCIO

FROM IMOVEL I

LEFT JOIN ANUNCIO A ON I.ID\_IMOVEL = A.ID\_IMOVEL;



Explicação: Lista todos os imóveis, mostrando NULL para os que não possuem anúncio.

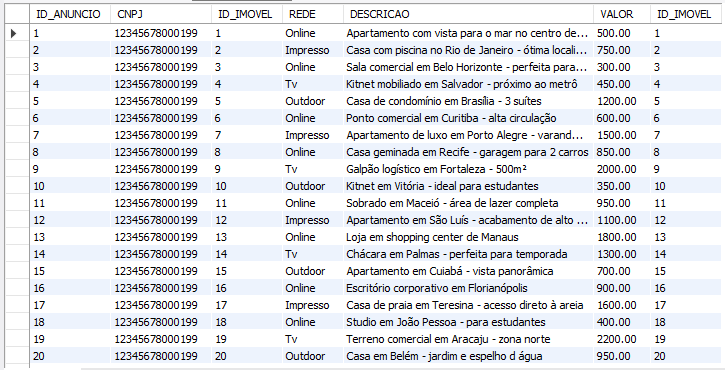
### 12.6 RIGHT JOIN: Todos os anúncios, mesmo sem imóvel

sql

SELECT A.\*, I.ID\_IMOVEL

FROM IMOVEL I

RIGHT JOIN ANUNCIO A ON I.ID\_IMOVEL = A.ID\_IMOVEL;



Explicação: Lista todos os anúncios, mostrando NULL para os que não possuem imóvel associado (possível inconsistência).

### 12.7 INNER JOIN: Funcionários com dados da empresa

sql

SELECT F.\*, E.NOME AS NOME\_EMPRESA

FROM FUNCIONARIO F

INNER JOIN EMPRESA E ON F.CNPJ = E.CNPJ;



Explicação: Retorna funcionários com informações da empresa.

### 12.8 LEFT JOIN: Reservas mesmo sem nota fiscal

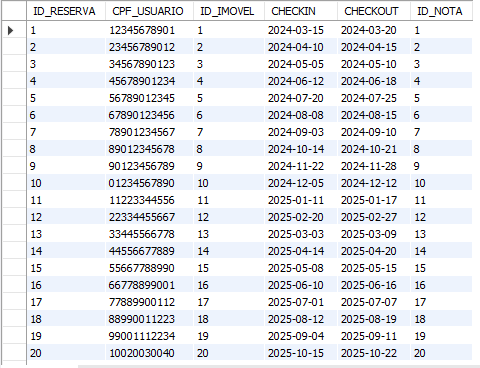
sql

SELECT R.\*, NF.ID\_NOTA

FROM RESERVA R

LEFT JOIN PAGAMENTO P ON R.ID\_RESERVA = P.ID\_RESERVA

LEFT JOIN NOTA\_FISCAL NF ON P.ID\_PAGAMENTO = NF.ID\_PAGAMENTO;



Explicação: Lista todas as reservas, mostrando NULL para as que não possuem nota fiscal.

### 12.9 RIGHT JOIN: Pagamentos sem reserva

sql

SELECT P.\*, R.ID\_RESERVA

FROM RESERVA R

RIGHT JOIN PAGAMENTO P ON R.ID\_RESERVA = P.ID\_RESERVA

WHERE R.ID\_RESERVA IS NULL;



Explicação: Identifica pagamentos sem reserva associada (possível inconsistência). Como não há, a tabela retorna vazia.

### 12.10 FULL JOIN simulado: Todos os usuários e proprietários

sql

SELECT U.CPF, U.NOME, 'Usuário' AS TIPO

FROM USUARIO U

LEFT JOIN PROPRIETARIO P ON U.CPF = P.CPF

WHERE P.CPF IS NULL

UNION ALL

SELECT P.CPF, P.NOME, 'Proprietário' AS TIPO

FROM PROPRIETARIO P

LEFT JOIN USUARIO U ON P.CPF = U.CPF

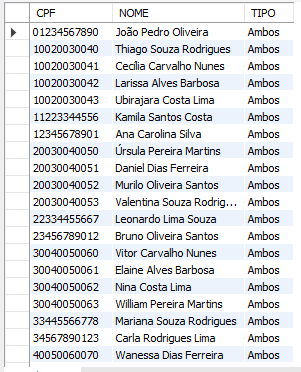
WHERE U.CPF IS NULL

UNION ALL

SELECT U.CPF, U.NOME, 'Ambos' AS TIPO

FROM USUARIO U

INNER JOIN PROPRIETARIO P ON U.CPF = P.CPF;



Explicação: Lista todas as pessoas cadastradas, classificando-as como usuários, proprietários ou ambos.

## 

## 

## 

## **13. Trigger e Stored Procedure**

### 13.1 Trigger: Validar data de check-out > check-in

sql

DELIMITER *//*

CREATE TRIGGER valida\_checkout\_checkin

BEFORE INSERT ON RESERVA

FOR EACH ROW

BEGIN

IF NEW.CHECKOUT <= NEW.CHECKIN THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Check-out deve ser após check-in';

END IF;

END;

*//*

DELIMITER ;



INSERT INTO RESERVA (CPF\_USUARIO, ID\_IMOVEL, CHECKIN, CHECKOUT) VALUES

('12345678901', 1, '2024-03-14', '2024-03-15');



INSERT INTO RESERVA (CPF\_USUARIO, ID\_IMOVEL, CHECKIN, CHECKOUT) VALUES

('12345678901', 1, '2024-03-25', '2024-03-30');



Explicação: Trigger que impede a inserção de reservas com data de check-out anterior ou igual ao check-in.

### 13.2 Stored Procedure: Inserir novo usuário

sql

DELIMITER *//*

CREATE PROCEDURE InserirUsuario(

IN p\_CPF VARCHAR(11),

IN p\_NOME VARCHAR(50),

IN p\_EMAIL VARCHAR(50),

IN p\_TELEFONE VARCHAR(15)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

START TRANSACTION;

IF EXISTS (SELECT 1 FROM USUARIO WHERE CPF = p\_CPF) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'CPF já cadastrado';

END IF;

IF EXISTS (SELECT 1 FROM USUARIO WHERE EMAIL = p\_EMAIL) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Email já cadastrado';

END IF;

INSERT INTO USUARIO (CPF, NOME, EMAIL, TELEFONE)

VALUES (p\_CPF, p\_NOME, p\_EMAIL, p\_TELEFONE);

COMMIT;

END;

*//*

DELIMITER ;



CALL InserirUsuario('17638406702','VICTHOR','VICTHORFELIX7@GMAIL.COM','21993701440');

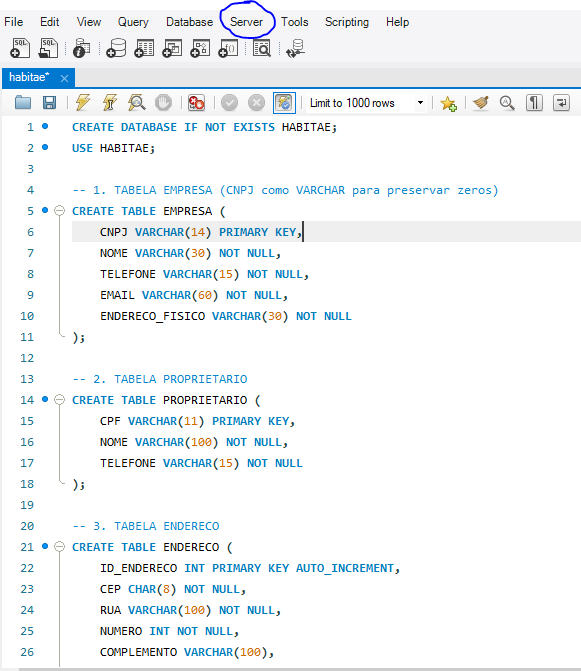


Explicação: Procedure para inserir novo usuário com validações de CPF e email únicos.

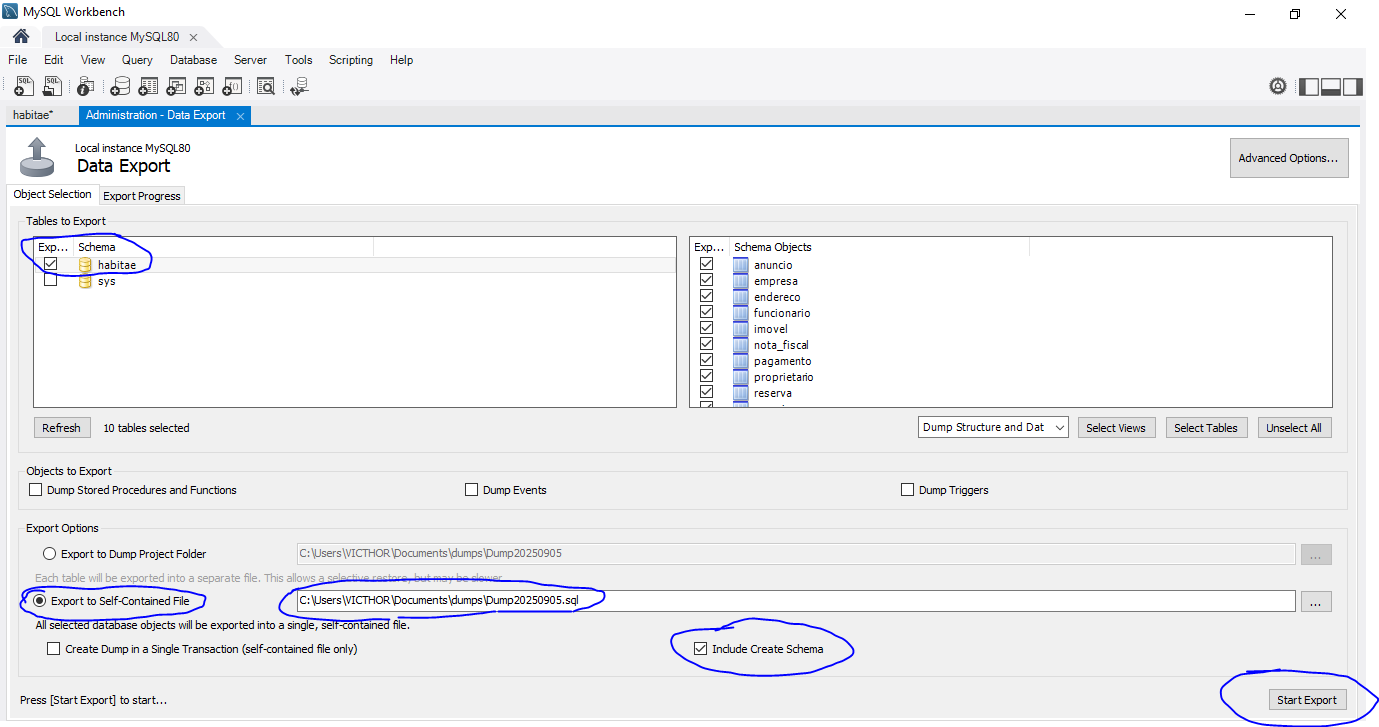
### **14. BACKUP**

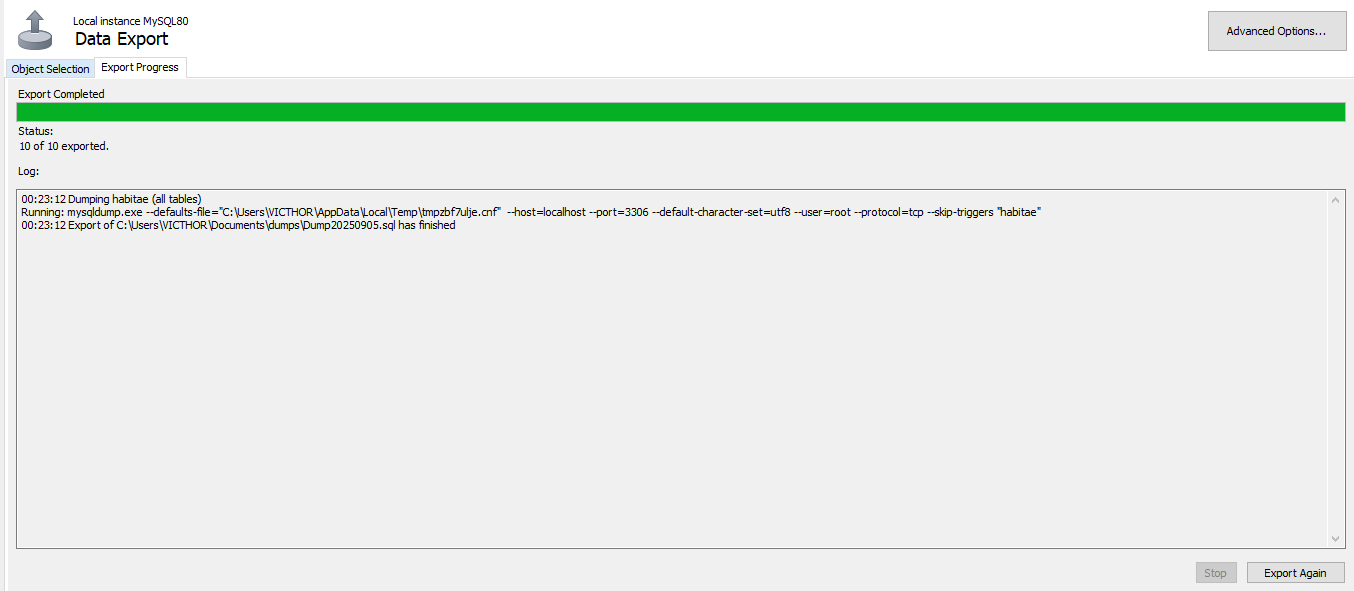
Processo de exportação e importação do backup do HABITAE.sql

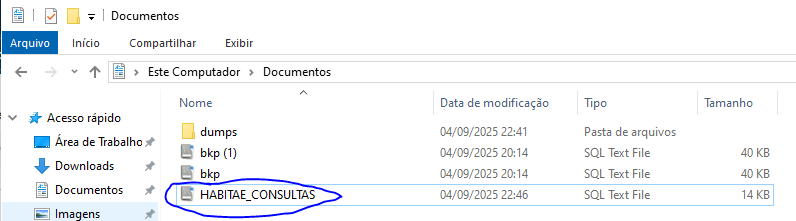
### 14.1 Exportação



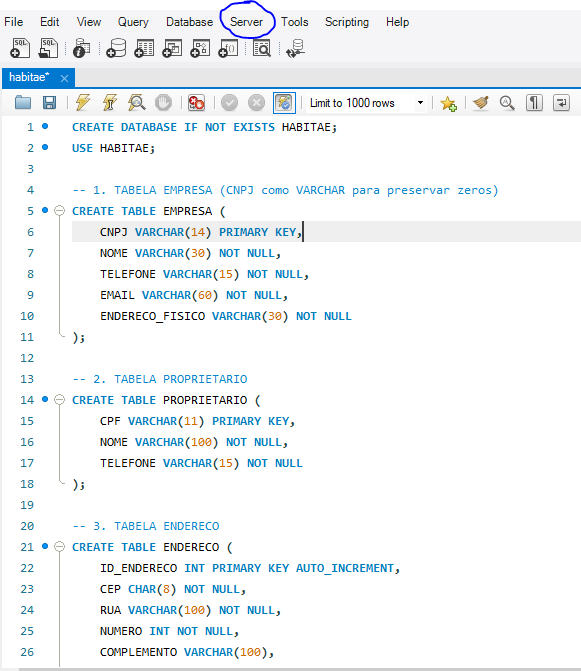
**\* Opção SERVER e após Data Export.**



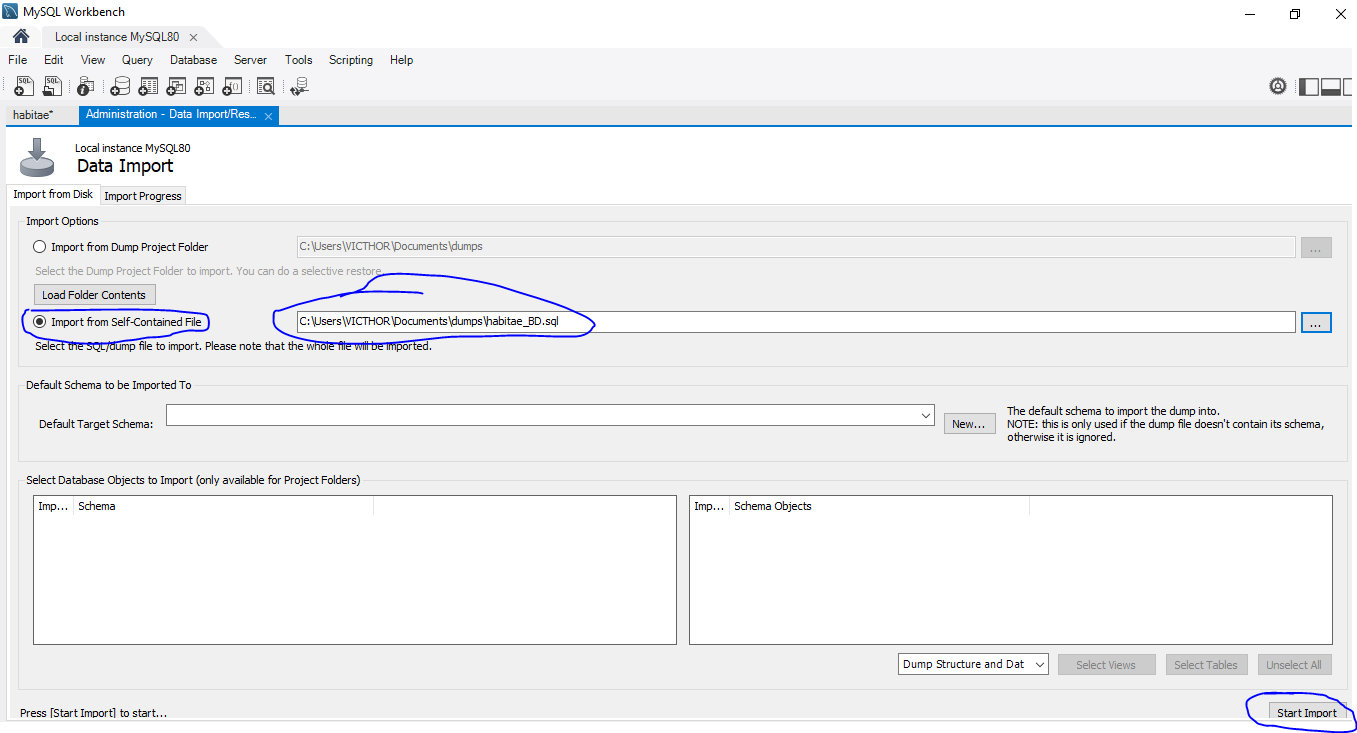


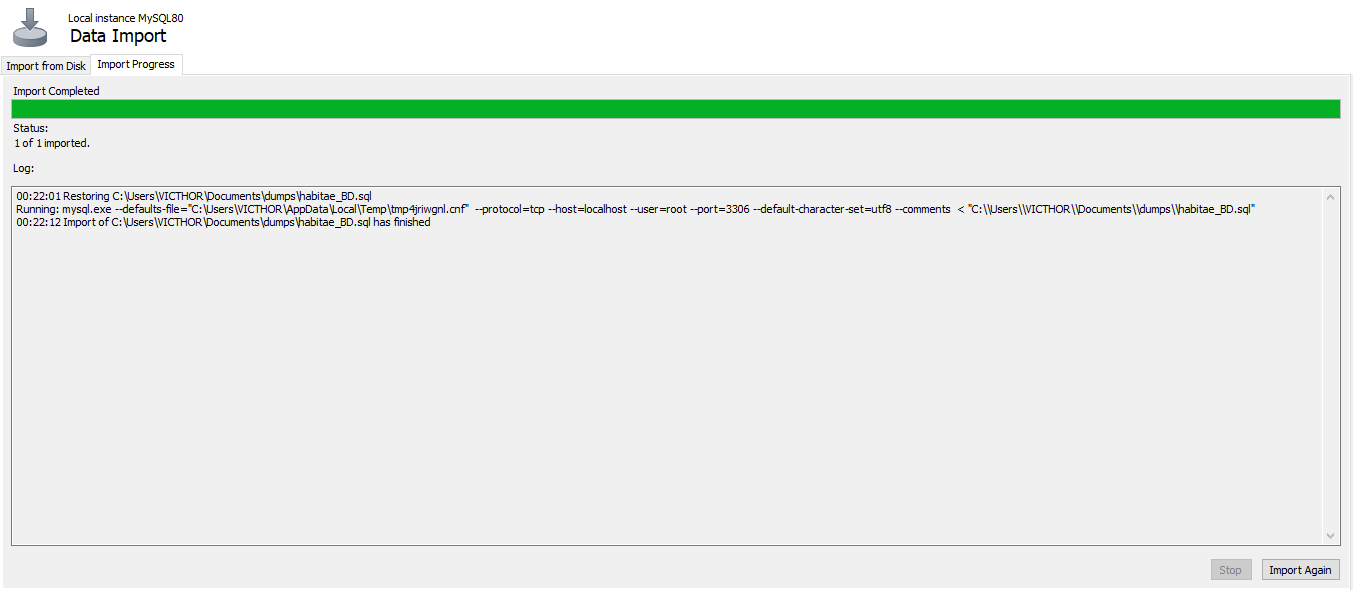


### 14.1 Importação



**\* Opção SERVER e após Data Import.**

****

****