**Consultas SQL**

**– Quais prédios que têm mais de 5 apartamentos? Group By/Having**

```
SELECT p.num_predio, COUNT(a.num_apartamento) AS total_apartamentos
FROM predios p
JOIN apartamentos a ON p.num_predio = a.num_predio
GROUP BY p.num_predio
HAVING COUNT(a.num_apartamento) > 5;
```

**– Quantos funcionários existem por prédios? (INNER JOIN)**

```
SELECT p.num_predio, p.quantidade_apartamentos, COUNT(f.cpf) AS total_funcionarios
FROM predios p
INNER JOIN funcionarios f ON p.num_predio = f.num_predio
GROUP BY p.num_predio, p.quantidade_apartamentos;
```

**-- Apartamento com a maior metragem dos predios**

```
SELECT num_predio, num_apartamento, metragem
FROM apartamentos A
WHERE metragem = (
   SELECT MAX(metragem)
   FROM apartamentos
);
```

**-- Quantidade de apartamentos cadastrados por prédios (ta ok)**

```
SELECT num_predio, COUNT(*) AS total_apartamentos
FROM apartamentos
group by num_predio;
```

```
-- Quantidade de apartamentos por prédio (ta ok)
select quantidade_apartamentos from predios;
```

```
-- Comodos por apartamento por prédio (ta ok)
SELECT A.num_predio, A.num_apartamento, A.num_comodos
FROM apartamentos A
Where A.num_predio IN (SELECT P.num_predio
                       FROM Predios P
                       WHERE P.quantidade_apartamentos > 1)
GROUP BY A.num_predio, A.num_apartamento, A.num_comodos;
```

**– TRIGGER –** colocando um unique no campo não precisa desse trigger

```
DELIMITER //

CREATE TRIGGER before_insert_sindico
BEFORE INSERT ON sindicos
FOR EACH ROW
BEGIN
   DECLARE num_sindicos INT;
   SET num_sindicos = (SELECT COUNT(*) FROM sindicos WHERE num_predio =
NEW.num_predio);

   IF num_sindicos > 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Este prédio já possui um
síndico.';
   END IF;
END;
//
```

**– PROCEDURE PARA INSERIR FUNCIONÁRIO SEM PRECISAR DIGITAR CPF DO SUPERVISOR JÁ CADASTRADO (PRECISA DE TESTE)**

**MYSQL**

```
DELIMITER //

CREATE PROCEDURE inserir_funcionario(
   IN p_cpf VARCHAR(11),
   IN p_cargo VARCHAR(20),
   IN p_salario FLOAT,
   IN p_cpf_supervisor VARCHAR(11),
   IN p_num_predio INT
)
BEGIN
   IF p_cpf_supervisor IS NULL OR LENGTH(p_cpf_supervisor) <> 11 OR p_cpf_supervisor
NOT REGEXP '^[0-9]+$' THEN
      INSERT INTO funcionarios (cpf, cargo, salario, cpf_supervisor, num_predio)
      SELECT p_cpf, p_cargo, p_salario, f.cpf, p_num_predio
      FROM funcionarios f
      WHERE f.num_predio = p_num_predio AND f.cpf = f.cpf_supervisor
      LIMIT 1;
   ELSE
      INSERT INTO funcionarios (cpf, cargo, salario, cpf_supervisor, num_predio)
      VALUES (p_cpf, p_cargo, p_salario, p_cpf_supervisor, p_num_predio);
   END IF;
END;
//

DELIMITER ;
```

**EX USO MYSQL:**

```
– CRIAR NOVA PESSOA
INSERT INTO pessoas (cpf, nome, email) VALUES ('77777777777', 'Deandra',
'deandra@example.com');

call inserir_funcionario(‘77777777777','Encanadora', 1700.00,0,1);
```

**ORACLE LIVE**

```
CREATE OR REPLACE PROCEDURE inserir_funcionario(
   p_cpf IN VARCHAR2,
   p_cargo IN VARCHAR2,
   p_salario IN NUMBER,
   p_cpf_supervisor IN VARCHAR2,
   p_num_predio IN NUMBER
)
IS
BEGIN
   IF p_cpf_supervisor IS NULL OR LENGTH(p_cpf_supervisor) <> 11 OR NOT
REGEXP_LIKE(p_cpf_supervisor, '^[0-9]+$') THEN
      INSERT INTO funcionarios (cpf, cargo, salario, cpf_supervisor, num_predio)
      SELECT p_cpf, p_cargo, p_salario, f.cpf, p_num_predio
      FROM funcionarios f
      WHERE f.num_predio = p_num_predio AND f.cpf = f.cpf_supervisor
      AND ROWNUM <= 1;
   ELSE
      INSERT INTO funcionarios (cpf, cargo, salario, cpf_supervisor, num_predio)
      VALUES (p_cpf, p_cargo, p_salario, p_cpf_supervisor, p_num_predio);
   END IF;
END;
/
```

**EX USO ORACLE LIVE:**

```
– CRIAR NOVA PESSOA
INSERT INTO pessoas (cpf, nome, email) VALUES ('77777777777', 'Deandra',
'deandra@example.com');

BEGIN
   inserir_funcionario(‘77777777777','Encanadora', 1700.00,0,1
);
END;
/
```

**-- (Uso da consulta escalar)**

```
Quantos condomínios existem na base de dados? --
select count(*)
from condominios;
```

Quantos prédios existem no condomínios?
select count(num_predio) from predios;

**-- (Uso do Group By)**
Quantos condomínios existem na base de dados e quais são seus endereços? --
select endereco, count(*)
from condominios
group by endereco;

Quantos visitantes cada apartamento recebeu – (ta ok)
select num_apartamento_informado, count(*)
from visitantes
group by num_apartamento_informado;

**-- (Uso do Group By) / Having**
Qual é o número de prédios que têm mais de 8 apartamentos?
select count(*), sum(quantidade_apartamentos)
from predios
group by num_predio
having sum(quantidade_apartamentos) > 8;

**- (Uso do where)**
Qual é o número de prédios que têm mais de 8 apartamentos?
SELECT  num_predio, COUNT(*)
FROM predios
WHERE quantidade_apartamentos > 5;

**-- (Uso de subconsulta por tabela)**
Quais são os nomes e emails dos proprietários dos apartamentos?
select p.nome, p.email
from pessoas p
where p.cpf in (select m.cpf from moradores m);

**-- (Uso LEFT JOIN)**
Quais são os apartamentos que não têm moradores?
select a.num_apartamento
from apartamentos a
left join moradores m on a.num_apartamento = m.num_apartamento
where m.num_apartamento is null;

-- Apartamentos não ocupados
SELECT num_apartamento
FROM Apartamentos
WHERE num_apartamento NOT IN (SELECT num_apartamento FROM Moradores);

```
-- (Procedure) Código para atualizar informação dos moradores
CREATE OR REPLACE PROCEDURE atualizar_dados_moradores (
   p_cpf VARCHAR2,
   p_nome VARCHAR2,
   p_email VARCHAR2,
   p_num_apartamento NUMBER
)+
AS
   v_num_moradores NUMBER;
BEGIN

   SELECT COUNT(*)
   INTO v_num_moradores
   FROM moradores
   WHERE cpf = p_cpf;

   IF v_num_moradores = 0 THEN
      DBMS_OUTPUT.PUT_LINE('CPF não encontrado.');
      RETURN;
   END IF;

   UPDATE pessoas
   SET nome = p_nome,
      email = p_email
   WHERE cpf = p_cpf;

   UPDATE moradores
   SET num_apartamento = p_num_apartamento
   WHERE cpf = p_cpf;

   COMMIT; -- Confirmar a transação
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      RAISE;
END;
/
```

**Maíra (consultas feitas)**

**-- quantidade de pessoas por apartamento (group by e having)**

**mysql**

```
SELECT a.num_apartamento, COUNT(m.cpf) AS QuantidadeMoradores
```

```
FROM apartamentos a
LEFT JOIN moradores m ON a.num_apartamento = m.num_apartamento
GROUP BY a.num_apartamento
HAVING QuantidadeMoradores > 0;
```

**oracle**

```
SELECT a.num_apartamento, COUNT(m.cpf) as quantidademoradores
FROM apartamentos a
LEFT JOIN moradores m ON a.num_apartamento = m.num_apartamento
GROUP BY a.num_apartamento
HAVING COUNT(m.cpf) > 0;
```

**– lista de moradores por apartamento (junçao interna)**

```
SELECT pessoas.nome, moradores.num_apartamento
FROM pessoas
INNER JOIN moradores ON moradores.cpf = pessoas.cpf;
```

**– lista de apartamentos com e sem moradores (junçao externa)**

```
SELECT apartamentos.num_apartamento,count(moradores.cpf)
FROM apartamentos
LEFT JOIN moradores ON apartamentos.num_apartamento = moradores.num_apartamento
GROUP BY apartamentos.num_apartamento;
```

**– apartamentos que tem pelo menos 1 morador (semi - junçao)**

```
SELECT num_apartamento
FROM apartamentos
WHERE num_apartamento IN (
   SELECT num_apartamento
   FROM moradores
);
```

**– Trigger e procedimento mysql**

```
DELIMITER //
CREATE TRIGGER trigger_verificar_limite_apartamentos
BEFORE INSERT ON apartamentos
FOR EACH ROW
BEGIN
   DECLARE total_apartamentos INT;
   DECLARE limite_apartamentos INT;

   -- Obtém o número total de apartamentos no prédio
   SELECT COUNT(*) INTO total_apartamentos
   FROM apartamentos
```

```
    WHERE num_predio = NEW.num_predio;

    SELECT quantidade_apartamentos INTO limite_apartamentos
    FROM predios
    WHERE num_predio = NEW.num_predio;

    -- Verifica se a adição do novo apartamento ultrapassa o limite
    IF total_apartamentos >= limite_apartamentos THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'A adição do novo apartamento ultrapassa o limite permitido
no prédio.';
    END IF;
END;
//
DELIMITER ;

DELIMITER //
```

**– trigger oracle**

```
CREATE OR REPLACE TRIGGER trigger_verificar_limite_apartamentos
BEFORE INSERT ON apartamentos
FOR EACH ROW
DECLARE
    total_apartamentos INT;
    limite_apartamentos INT;
BEGIN
    SELECT COUNT(*) INTO total_apartamentos
    FROM apartamentos
    WHERE num_predio = :NEW.num_predio;

    SELECT quantidade_apartamentos INTO limite_apartamentos
    FROM predios
    WHERE num_predio = :NEW.num_predio;

    -- Verifica se a adição do novo apartamento ultrapassa o limite
    IF total_apartamentos >= limite_apartamentos THEN
        RAISE_APPLICATION_ERROR(-20001, 'A adição do novo apartamento ultrapassa o
limite permitido no prédio.');
    END IF;
END;
/
```

**– procedimento mysql**

```
DELIMITER
CREATE PROCEDURE atualizar_salario_funcionario(
    funcionario_cpf VARCHAR(11),
    novo_salario FLOAT
)
```

```
BEGIN
    IF NOT EXISTS (SELECT 1 FROM funcionarios WHERE cpf = funcionario_cpf) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Funcionário não encontrado.';
    END IF;

    UPDATE funcionarios
    SET salario = novo_salario
    WHERE cpf = funcionario_cpf;

    SELECT 'Salário do funcionário atualizado com sucesso.' AS Message;
END //

DELIMITER ;

CALL atualizar_salario_funcionario('98765432109', 2800.00);
```

**– procedimento em oracle parcialmente certo**

```
CREATE OR REPLACE PROCEDURE atualizar_salario_funcionario(
    funcionario_cpf VARCHAR2,
    novo_salario FLOAT
)
IS
BEGIN

    UPDATE funcionarios
    SET salario = novo_salario
    WHERE cpf = funcionario_cpf;

            COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK; -- Desfaz a transação em caso de erro
        RAISE_APPLICATION_ERROR(-20001, 'Erro ao atualizar o salário do funcionário.');
END;
```

**-- lista do nomes de sindico**

```
SELECT pessoas.nome, sindicos.num_predio
FROM pessoas
INNER JOIN sindicos ON sindicos.cpf = pessoas.cpf;
```

**-- lista de quais funcionarios trabalham em quais predios**

```
SELECT pessoas.nome, funcionarios.num_predio
FROM pessoas
INNER JOIN funcionarios ON funcionarios.cpf = pessoas.cpf;
```

**-- lista de quais funcionarios trabalham no predio 1**

```
SELECT pessoas.nome, funcionarios.num_predio
FROM pessoas
inner JOIN funcionarios ON pessoas.cpf = funcionarios.cpf
WHERE funcionarios.num_predio = 1;
```

**-- lista de reservas de áreas comuns**

```
select * from reservas;
```

**-- lista  dos apartamentos que reservaram as áreas comuns**

```
select num_apartamento, COUNT(num_apartamento) as QuantidadeDeReservas
From reservas
group by num_apartamento
Order by QuantidadeDeReservas Desc;

-- quais os nomes dos funcionarios que prestaram serviços para os apartamentos

select pessoas.nome
from pessoas
where pessoas.CPF in (select CPF from servicos);

-- quantas vezes o funcionario prestou serviço para um apartamento

select cpf, count(cpf), num_apartamento
from servicos
group by cpf, num_apartamento;
```

**–Número de cômodos por apartamento por predio**

```
SUBCONSULTA
select  A.num_comodos
from ApartamentoS A
WHERE A.num_predio = (SELECT P.num_predio
                                        From Predios P
                                        where A.num_predio = P.num_predio);
```

JOIN

```
SELECT P.num_predio, A.num_apartamento, A.num_comodos
FROM predios P
```

```
JOIN apartamentos A ON P.num_predio = A.num_predio
WHERE P.quantidade_apartamentos > 1;
```

**–Informações moradores**

JOIN

```
SELECT p.nome, p.email, m.cpf
from Pessoas P
JOIN Moradores M ON p.cpf = m.cpf
where m.num_apartamento = '101';
```

SUBCONSULTA

```
SELECT  m.cpf
from moradores m
Where m.cpf in (SELECT p.cpf
                        From Pessoas p
                        Where nome = 'Ana Santos')
```

**–Apartamentos por condomínio**

```
SELECT c.endereco, COUNT(p.num_predio) AS total_apartamentos
FROM condominios c
LEFT JOIN predios p ON c.endereco = p.endereco
GROUP BY c.endereco;
```

```
SELECT count(a.apartamentos)
from Apartamentos A,
Where A.endereço = (SELECT C.endereço
                        From Condominios C
                        Where A.endereco = C.endereco)
```

**–Funcionários e seus respectivos supervisores**

```
SELECT f.cpf, f.cargo, f.salario, f.cpf_supervisor, s.nome AS nome_supervisor
FROM funcionarios f
LEFT JOIN pessoas s ON f.cpf_supervisor = s.cpf;
```

```
SELECT F.CPF, F.CARGO, F.CPF_SUPERVISOR
FROM FUNCIONARIOS F
```

**–Número de visitas por apartamento**

```
SELECT v.num_apartamento_informado, COUNT(v.cpf) AS total_visitantes
```

```
FROM visitanteS v
GROUP BY v.num_apartamento_informado;
```

**–Serviços contratados por apartamento**

```
SELECT c.num_apartamento, c.cpf AS cpf_funcionario, s.descricao_servico, e.cnpj
FROM contrata c
JOIN servicos s ON c.num_apartamento = s.num_apartamento AND c.cpf = s.cpf
JOIN empresas_servicos e ON c.cnpj = e.cnpj;
```

**–Lista de síndicos e seus respectivos prédios**

```
SELECT s.cpf, s.profissao, p.num_predio
FROM sindicos s
JOIN predios p ON s.num_predio = p.num_predio;
```

**–Visitas por apartamento em determinada data**

```
SELECT v.num_apartamento, v.data_visita, COUNT(v.cpf) AS total_visitas
FROM visitas v
WHERE v.data_visita = TO_DATE('2024-03-10', 'YYYY-MM-DD')
GROUP BY v.num_apartamento, v.data_visita;
```

**–Telefones dos moradores**

```
SELECT m.num_apartamento, t.telefone
FROM moradores m
JOIN telefones t ON m.cpf = t.cpf
WHERE m.num_apartamento IN (SELECT num_apartamento FROM apartamentos WHERE
num_predio = 1);
```

```
SELECT *
FROM telefones t
WHERE telefone is not null
and t.cpf = (SELECT m.cpf
             FROM moradores m
             WHERE m.cpf = t.cpf)
```

**–Reservas de áreas comuns por tipo e data**

```
SELECT r.endereco, r.tipo, r.data_reserva, COUNT(r.num_apartamento) AS total_reservas
FROM reservas r
GROUP BY r.endereco, r.tipo, r.data_reserva;
```

**–Contratação de serviços de empresa específica**

```
SELECT c.num_apartamento, c.cpf, c.data_servico, c.horario_servico
FROM contrata c
```

```
WHERE c.cnpj = '12345678901234';


--Visitantes que informam número de prédio específico
SELECT v.cpf, v.num_apartamento_informado, v.num_predio_informado, p.nome AS
nome_visitante
FROM visitantes v
JOIN pessoas p ON v.cpf = p.cpf
WHERE v.num_predio_informado = 1;


--Horários e datas de visita por apartamento
SELECT v.num_apartamento, v.cpf, v.data_visita, v.horario_visita
FROM visitas v
ORDER BY v.num_apartamento, v.data_visita, v.horario_visita;

--Serviços realizados por um funcionário em um apartamento
SELECT s.num_apartamento, s.data_servico, s.horario_servico, s.descricao_servico
FROM servicos s
WHERE s.cpf = '12345678901' AND s.num_apartamento = 101;


--Pessoas que não são nem moradores nem funcionários
SELECT cpf, nome, email
FROM pessoas
WHERE cpf NOT IN (SELECT cpf FROM moradores)
AND cpf NOT IN (SELECT cpf FROM funcionarios);

--Apartamentos e áreas comuns
SELECT a.num_apartamento, ac.endereco, ac.tipo
FROM apartamentos a
LEFT JOIN areas_comuns ac ON a.num_predio = ac.endereco;

--Funcionários que são supervisores
SELECT f.cpf, f.nome, f.cargo, f.cpf_supervisor
FROM funcionarios f
WHERE f.cpf = f.cpf_supervisor;

--Apartamentos sem reservas em áreas comuns
SELECT a.num_apartamento
FROM apartamentos a
LEFT JOIN reservas r ON a.num_apartamento = r.num_apartamento
WHERE r.num_apartamento IS NULL;


SELECT a.num_apartamento
from apartamentos a
```

```
where a.num_apartamento =( select r.num_apartamento
                           from reserva r
                           where r.num_apartamento is null)

--Média salarial dos funcionários por cargo
SELECT cargo, AVG(salario) AS media_salarial
FROM funcionarios
GROUP BY cargo;

--Total de visitantes por dia
SELECT data_visita, COUNT(cpf) AS total_visitantes
FROM visitas
GROUP BY data_visita;

--Empresas que prestaram serviço em determinado apartamento
SELECT c.num_apartamento, e.cnpj
FROM contrata c
JOIN empresas_servicos e ON c.cnpj = e.cnpj
WHERE c.num_apartamento = 101;

--Apartamentos que contratam serviços em data específica
SELECT c.num_apartamento, s.descricao_servico
FROM contrata c
JOIN servicos s ON c.num_apartamento = s.num_apartamento
WHERE c.data_servico = TO_DATE('2024-03-15', 'YYYY-MM-DD');

--Número de visitas por apartamento
SELECT v.num_apartamento, COUNT(v.cod_registro) AS total_registros
FROM visitas v
GROUP BY v.num_apartamento;


--Visitantes que estiveram em um apartamento específico
SELECT v.cpf, p.nome AS nome_visitante, v.data_visita, v.horario_visita
FROM visitas v
JOIN pessoas p ON v.cpf = p.cpf
WHERE v.num_apartamento = 101;

--Apartamentos e todas as suas reservas feitas
SELECT r.num_apartamento, r.endereco, r.tipo, r.data_reserva, r.horario_reserva
FROM reservas r
JOIN apartamentos a ON r.num_apartamento = a.num_apartamento;

--Total de apartamentos por número de cômodos (quantos apartamentos tem com
quantidade x de cômodos)
SELECT num_comodos, COUNT(num_apartamento) AS total_apartamentos
FROM apartamentos
GROUP BY num_comodos;
```

**–Funcionários que não são supervisores**

```
SELECT f.cpf, f.nome, f.cargo
FROM funcionarios f
WHERE f.cpf NOT IN (SELECT cpf_supervisor FROM funcionarios WHERE cpf_supervisor
IS NOT NULL);
```

**–Data e hora de todas as visitas registradas**

```
SELECT data_visita, horario_visita
FROM visitas;
```

**–Funcionários que trabalham em um prédio específico**

```
SELECT f.cpf, f.nome, f.cargo, p.num_predio
FROM funcionarios f
JOIN predios p ON f.num_predio = p.num_predio
WHERE p.num_predio = 1;
```

**–Apartamentos com moradores e seus telefones**

```
SELECT m.num_apartamento, m.cpf AS cpf_morador, p.nome AS nome_morador, t.telefone
FROM moradores m
JOIN pessoas p ON m.cpf = p.cpf
JOIN telefones t ON m.cpf = t.cpf;

SELECT m.cpf
FROM Moradores m
Where m.cpf in (select  t.cpf
                from telefones t
                where telefone is not null)
```

**–Áreas comuns sem reservas em determinada data**

```
SELECT ac.endereco, ac.tipo
FROM areas_comuns ac
LEFT JOIN reservas r ON ac.endereco = r.endereco AND ac.tipo = r.tipo AND
r.data_reserva = TO_DATE('2024-03-10', 'YYYY-MM-DD')
WHERE r.endereco IS NULL;
```

**–Apartamentos sem visitantes em uma determinada data**

```
SELECT a.num_apartamento
FROM apartamentos a
LEFT JOIN visitas v ON a.num_apartamento = v.num_apartamento AND v.data_visita =
TO_DATE('2024-03-10', 'YYYY-MM-DD')
WHERE v.num_apartamento IS NULL;
```

**–Funcionários por cargo(total)**

```
SELECT cargo, COUNT(cpf) AS total_funcionarios
```

```
FROM funcionarios
GROUP BY cargo;


--Média de metragem dos apartamentos por número de cômodos
SELECT A.num_comodos, AVG(A.metragem) as media_metragem
FROM Apartamentos A
Group by A.num_comodos
```