```
--
**********************************************************************
***************
--                                  REVISÃO P1
--
**********************************************************************
***************

--
**********************************************************************
***************
--                                  TAREFA 01
--
**********************************************************************
***************

-- funcionario
CREATE TABLE Funcionario (
id_funcionario INT NOT NULL PRIMARY KEY,
nm_funcionario VARCHAR2(60) NOT NULL
);

-- inserindo registro no funcionario
INSERT INTO Funcionario (id_funcionario, nm_funcionario)
VALUES (1, 'João Silva');
INSERT INTO Funcionario (id_funcionario, nm_funcionario)
VALUES (2, 'Maria Oliveira');
INSERT INTO Funcionario (id_funcionario, nm_funcionario)
VALUES (3, 'Pedro Santos');
INSERT INTO Funcionario (id_funcionario, nm_funcionario)
VALUES (4, 'Ana Pereira');
INSERT INTO Funcionario (id_funcionario, nm_funcionario)
VALUES (5, 'Carlos Souza');

-- pedidos
CREATE TABLE Pedidos(
id_pedido NUMBER PRIMARY KEY,
```

```
dt_pedido DATE,
nm_prato VARCHAR2(60 CHAR) NOT NULL,
nm_bebida VARCHAR2(60 CHAR) NOT NULL,
id_funcionario NUMBER NOT NULL,
CONSTRAINT FK_Funcionario FOREIGN KEY (id_funcionario)
REFERENCES Funcionario(id_funcionario)
);

-- inserindo registros nos Pedidos

INSERT INTO Pedidos (id_pedido, dt_pedido, nm_prato,
nm_bebida, id_funcionario)
VALUES (1, TO_DATE('2024-03-12', 'YYYY-MM-DD'), 'Pizza
Margherita', 'Coca-Cola', 1);
INSERT INTO Pedidos (id_pedido, dt_pedido, nm_prato,
nm_bebida, id_funcionario)
VALUES (2, TO_DATE('2024-03-12', 'YYYY-MM-DD'), 'Hambúrguer',
'Suco de Laranja', 2);
INSERT INTO Pedidos (id_pedido, dt_pedido, nm_prato,
nm_bebida, id_funcionario)
VALUES (3, TO_DATE('2024-03-12', 'YYYY-MM-DD'), 'Salada
Caesar', 'Água Mineral', 3);
INSERT INTO Pedidos (id_pedido, dt_pedido, nm_prato,
nm_bebida, id_funcionario)
VALUES (4, TO_DATE('2024-03-12', 'YYYY-MM-DD'), 'Sushi Combo',
'Chá Verde', 4);
INSERT INTO Pedidos (id_pedido, dt_pedido, nm_prato,
nm_bebida, id_funcionario)
VALUES (5, TO_DATE('2024-03-12', 'YYYY-MM-DD'), 'Lasanha',
'Vinho Tinto', 5);

-- mesa
CREATE TABLE Mesa (
ID_numeroMesa INT PRIMARY KEY,
qt_numeroLugar NUMBER,
ds_localizacao VARCHAR(50),
id_pedido NUMBER,
```

```
FOREIGN KEY (id_pedido) REFERENCES Pedidos (id_pedido)
);

-- inserindo Registros da Mesa
INSERT INTO Mesa (ID_numeroMesa, qt_numeroLugar,
ds_localizacao, id_pedido)
VALUES (1, 4, 'Perto da janela', 1);

INSERT INTO Mesa (ID_numeroMesa, qt_numeroLugar,
ds_localizacao, id_pedido)
VALUES (2, 6, 'Na área externa', 2);

INSERT INTO Mesa (ID_numeroMesa, qt_numeroLugar,
ds_localizacao, id_pedido)
VALUES (3, 2, 'Ao lado do bar', 3);

INSERT INTO Mesa (ID_numeroMesa, qt_numeroLugar,
ds_localizacao, id_pedido)
VALUES (4, 8, 'No centro do restaurante', 4);

INSERT INTO Mesa (ID_numeroMesa, qt_numeroLugar,
ds_localizacao, id_pedido)
VALUES (5, 4, 'Na área de fumantes', 5);

-- clientes
CREATE TABLE Clientes (
id_cliente NUMBER PRIMARY KEY,
nm_cliente VARCHAR2(60 CHAR) NOT NULL,
dt_nascimento_cliente DATE,
tel_cliente NUMBER NOT NULL,
nm_email_cliente VARCHAR2(100 CHAR) NOT NULL,
cd_cpf_cliente VARCHAR2(20) NOT NULL,
cd_cep_cliente VARCHAR2(20) NOT NULL,
num_endereco_cliente NUMBER NOT NULL,
nm_complemento_cliente VARCHAR2(100 CHAR),
id_pedido NUMBER,
id_numeroMesa NUMBER,
```

```
CONSTRAINT fk_pedido FOREIGN KEY (id_pedido) REFERENCES
Pedidos (id_pedido),
CONSTRAINT fk_Mesa FOREIGN KEY (id_numeroMesa) REFERENCES Mesa
(ID_numeroMesa)
);

-- inserindo na tabela Clientes
INSERT INTO Clientes (id_cliente, nm_cliente,
dt_nascimento_cliente, tel_cliente,
nm_email_cliente, cd_cpf_cliente, cd_cep_cliente,
num_endereco_cliente,
nm_complemento_cliente, id_pedido, id_numeroMesa)
VALUES (1, 'Fernanda Oliveira', TO_DATE('1990-05-15',
'YYYY-MM-DD'), 1234567890,
'fernanda@example.com', '123.456.789-00', '12345-678', 123,
'Apartamento 101', 1, 1);

INSERT INTO Clientes (id_cliente, nm_cliente,
dt_nascimento_cliente, tel_cliente,
nm_email_cliente, cd_cpf_cliente, cd_cep_cliente,
num_endereco_cliente,
nm_complemento_cliente, id_pedido, id_numeroMesa)
VALUES (2, 'Ricardo Silva', TO_DATE('1985-08-20',
'YYYY-MM-DD'), 987654321,
'ricardo@example.com', '987.654.321-00', '98765-432', 456,
'Sala 2', 2, 2);

INSERT INTO Clientes (id_cliente, nm_cliente,
dt_nascimento_cliente, tel_cliente,
nm_email_cliente, cd_cpf_cliente, cd_cep_cliente,
num_endereco_cliente,
nm_complemento_cliente, id_pedido, id_numeroMesa)
VALUES (3, 'Amanda Souza', TO_DATE('1992-03-10',
'YYYY-MM-DD'), 456123789,
'amanda@example.com', '456.123.789-00', '45678-912', 789,
NULL, 3, 3);
```

```sql
INSERT INTO Clientes (id_cliente, nm_cliente,
dt_nascimento_cliente, tel_cliente,
nm_email_cliente, cd_cpf_cliente, cd_cep_cliente,
num_endereco_cliente,
nm_complemento_cliente, id_pedido, id_numeroMesa)
VALUES (4, 'Lucas Santos', TO_DATE('1988-11-25',
'YYYY-MM-DD'), 789123456,
'lucas@example.com', '789.123.456-00', '78901-234', 1011,
'Bloco A', 4, 4);

INSERT INTO Clientes (id_cliente, nm_cliente,
dt_nascimento_cliente, tel_cliente,
nm_email_cliente, cd_cpf_cliente, cd_cep_cliente,
num_endereco_cliente,
nm_complemento_cliente, id_pedido, id_numeroMesa)
VALUES (5, 'Juliana Pereira', TO_DATE('1995-06-30',
'YYYY-MM-DD'), 321654987,
'juliana@example.com', '321.654.987-00', '32109-876', 1213,
'Casa 3', 5, 5);

-- tipo Ingrediente
CREATE TABLE TIPO_INGREDIENTE(
cd_tipo_ingrediente INTEGER PRIMARY KEY,
nm_tipo_ingrediente VARCHAR(30) NOT NULL
)

-- insert na tabela Tipo Ingrediente
INSERT INTO TIPO_INGREDIENTE (cd_tipo_ingrediente,
nm_tipo_ingrediente) VALUES (1, 'Grãos');

INSERT INTO TIPO_INGREDIENTE (cd_tipo_ingrediente,
nm_tipo_ingrediente) VALUES (2, 'Vegetais');

INSERT INTO TIPO_INGREDIENTE (cd_tipo_ingrediente,
nm_tipo_ingrediente) VALUES (3, 'Frutas');
```

```sql
INSERT INTO TIPO_INGREDIENTE (cd_tipo_ingrediente,
nm_tipo_ingrediente) VALUES (4, 'Carnes');

INSERT INTO TIPO_INGREDIENTE (cd_tipo_ingrediente,
nm_tipo_ingrediente) VALUES (5, 'Laticínios');

-- fornecedor
CREATE TABLE FORNECEDOR (
cd_fornecedor INT NOT NULL,
nm_fornecedor VARCHAR2(100) NOT NULL,
cd_telefone_fornecedor VARCHAR2(20) NOT NULL,
cd_cnpj VARCHAR2(14) NOT NULL, constraint FORNECEDOR_PK
PRIMARY KEY (cd_fornecedor)
);

-- insert na tabela Fornecedor
INSERT INTO FORNECEDOR (cd_fornecedor, nm_fornecedor,
cd_telefone_fornecedor, cd_cnpj)
VALUES (1, 'Fornecedor A', '1234567890', '12345678901234');

INSERT INTO FORNECEDOR (cd_fornecedor, nm_fornecedor,
cd_telefone_fornecedor, cd_cnpj)
VALUES (2, 'Fornecedor B', '0987654321', '98765432109876');

INSERT INTO FORNECEDOR (cd_fornecedor, nm_fornecedor,
cd_telefone_fornecedor, cd_cnpj)
VALUES (3, 'Fornecedor C', '1112223334', '111222333444455');

INSERT INTO FORNECEDOR (cd_fornecedor, nm_fornecedor,
cd_telefone_fornecedor, cd_cnpj)
VALUES (4, 'Fornecedor D', '4445556667', '44455566677788');

INSERT INTO FORNECEDOR (cd_fornecedor, nm_fornecedor,
cd_telefone_fornecedor, cd_cnpj)
VALUES (5, 'Fornecedor E', '7778889990', '77788899900011');

-- ingredientes
```

```sql
CREATE TABLE INGREDIENTE (
cd_ingrediente INTEGER PRIMARY KEY,
nm_ingrediente VARCHAR2(20) NOT NULL,
dt_validade_ingrediente DATE NOT NULL,
qt_ingrediente INTEGER NOT NULL,
dt_recebimento_ingrediente DATE NOT NULL,
fk_tipo_ingrediente INTEGER NOT NULL,
fk_fornecedor INTEGER NOT NULL,
cd_tipo_ingrediente NUMBER,
cd_fornecedor NUMBER,
CONSTRAINT fk_tipo_ingrediente FOREIGN KEY
(fk_tipo_ingrediente) REFERENCES TIPO_INGREDIENTE
(cd_tipo_ingrediente),
CONSTRAINT fk_fornecedor FOREIGN KEY (fk_fornecedor)
REFERENCES FORNECEDOR (cd_fornecedor)
);

-- inserindo na tabela Ingredientes
INSERT INTO INGREDIENTE (cd_ingrediente, nm_ingrediente,
dt_validade_ingrediente,
qt_ingrediente, dt_recebimento_ingrediente,
fk_tipo_ingrediente, fk_fornecedor)
VALUES (1, 'Farinha de Trigo', TO_DATE('2024-12-31',
'YYYY-MM-DD'), 100, TO_DATE('2024-03-12',
'YYYY-MM-DD'), 1, 1);

INSERT INTO INGREDIENTE (cd_ingrediente, nm_ingrediente,
dt_validade_ingrediente,
qt_ingrediente, dt_recebimento_ingrediente,
fk_tipo_ingrediente, fk_fornecedor)
VALUES (2, 'Tomate', TO_DATE('2024-03-15', 'YYYY-MM-DD'), 50,
TO_DATE('2024-03-10', 'YYYY-MMDD'), 2, 2);

INSERT INTO INGREDIENTE (cd_ingrediente, nm_ingrediente,
dt_validade_ingrediente,
qt_ingrediente, dt_recebimento_ingrediente,
fk_tipo_ingrediente, fk_fornecedor)
```

```
VALUES (3, 'Maçã', TO_DATE('2024-03-20', 'YYYY-MM-DD'), 30,
TO_DATE('2024-03-08', 'YYYY-MMDD'), 3, 3);

INSERT INTO INGREDIENTE (cd_ingrediente, nm_ingrediente,
dt_validade_ingrediente,
qt_ingrediente, dt_recebimento_ingrediente,
fk_tipo_ingrediente, fk_fornecedor)
VALUES (4, 'Carne Bovina', TO_DATE('2024-03-25',
'YYYY-MM-DD'), 80, TO_DATE('2024-03-05', 'YYYY-MM-DD'), 4, 4);

INSERT INTO INGREDIENTE (cd_ingrediente, nm_ingrediente,
dt_validade_ingrediente,
qt_ingrediente, dt_recebimento_ingrediente,
fk_tipo_ingrediente, fk_fornecedor)
VALUES (5, 'Queijo Mussarela', TO_DATE('2024-03-18',
'YYYY-MM-DD'), 40, TO_DATE('2024-03-03', 'YYYY-MM-DD'), 5, 5);

-- pedido Fornecedor
CREATE TABLE PEDIDO_FORNECEDOR (
cd_pedido_fornecedor INT NOT NULL,
dt_pedido_fornecedor TIMESTAMP NOT NULL,
vl_total_pedido_fornecedor DECIMAL NOT NULL,
ds_observacao_pedido_fornecedor VARCHAR2(255) NOT NULL,
cd_nfe_pedido_fornecedor VARCHAR2(50) NOT NULL,
dt_entrega_pedido_fornecedor TIMESTAMP NOT NULL,
cd_fornecedor INT NOT NULL,
constraint PEDIDO_FORNECEDOR_PK PRIMARY KEY
(cd_pedido_fornecedor)
);

-- inserir na tabela Pedido Fornecedor
INSERT INTO PEDIDO_FORNECEDOR (cd_pedido_fornecedor,
dt_pedido_fornecedor,
vl_total_pedido_fornecedor, ds_observacao_pedido_fornecedor,
cd_nfe_pedido_fornecedor,
dt_entrega_pedido_fornecedor, cd_fornecedor)
```

```
VALUES (1, CURRENT_TIMESTAMP, 150.50, 'Urgente: Entregar o
mais rápido possível.', '123456789', CURRENT_TIMESTAMP, 1);

INSERT INTO PEDIDO_FORNECEDOR (cd_pedido_fornecedor,
dt_pedido_fornecedor,
vl_total_pedido_fornecedor, ds_observacao_pedido_fornecedor,
cd_nfe_pedido_fornecedor,
dt_entrega_pedido_fornecedor, cd_fornecedor)
VALUES (2, CURRENT_TIMESTAMP, 200.00, 'Favor entregar na
portaria.', '987654321', CURRENT_TIMESTAMP, 2);

INSERT INTO PEDIDO_FORNECEDOR (cd_pedido_fornecedor,
dt_pedido_fornecedor,
vl_total_pedido_fornecedor, ds_observacao_pedido_fornecedor,
cd_nfe_pedido_fornecedor,
dt_entrega_pedido_fornecedor, cd_fornecedor)
VALUES (3, CURRENT_TIMESTAMP, 300.75, 'Favor ligar antes de
entregar.', '111222333', CURRENT_TIMESTAMP, 3);

INSERT INTO PEDIDO_FORNECEDOR (cd_pedido_fornecedor,
dt_pedido_fornecedor,
vl_total_pedido_fornecedor, ds_observacao_pedido_fornecedor,
cd_nfe_pedido_fornecedor,
dt_entrega_pedido_fornecedor, cd_fornecedor)
VALUES (4, CURRENT_TIMESTAMP, 180.25, 'Entregar apenas de
manhã.', '444555666', CURRENT_TIMESTAMP, 4);

INSERT INTO PEDIDO_FORNECEDOR (cd_pedido_fornecedor,
dt_pedido_fornecedor,
vl_total_pedido_fornecedor, ds_observacao_pedido_fornecedor,
cd_nfe_pedido_fornecedor,
dt_entrega_pedido_fornecedor, cd_fornecedor)
VALUES (5, CURRENT_TIMESTAMP, 250.00, 'Entregar na recepção.',
'777888999', CURRENT_TIMESTAMP, 5);

-- pedido Forcenedor Cliente
CREATE TABLE PEDIDO_FORNECEDOR_INGREDIENTE (
```

```
cd_pedido_fornecedor_ingrediente INT NOT NULL,
cd_pedido_fornecedor INT NOT NULL,
qt_pedido_fornecedor_ingrediente FLOAT NOT NULL,
vl_unitario_pedido_fornecedor_ingrediente DECIMAL NOT NULL,
id_ingrediente INT NOT NULL,
constraint PEDIDO_FORNECEDOR_INGREDIENTE_PK PRIMARY KEY
(cd_pedido_fornecedor_ingrediente)
);

-- inserir na tabela Pedido Fornecedor Cliente
INSERT INTO PEDIDO_FORNECEDOR_INGREDIENTE
(cd_pedido_fornecedor_ingrediente,
cd_pedido_fornecedor, qt_pedido_fornecedor_ingrediente,
vl_unitario_pedido_fornecedor_ingrediente, id_ingrediente)
VALUES (1, 1, 10.0, 5.0, 1);

INSERT INTO PEDIDO_FORNECEDOR_INGREDIENTE
(cd_pedido_fornecedor_ingrediente,
cd_pedido_fornecedor, qt_pedido_fornecedor_ingrediente,
vl_unitario_pedido_fornecedor_ingrediente, id_ingrediente)
VALUES (2, 2, 8.0, 4.0, 2);

INSERT INTO PEDIDO_FORNECEDOR_INGREDIENTE
(cd_pedido_fornecedor_ingrediente,
cd_pedido_fornecedor, qt_pedido_fornecedor_ingrediente,
vl_unitario_pedido_fornecedor_ingrediente, id_ingrediente)
VALUES (3, 3, 15.0, 3.0, 3);

INSERT INTO PEDIDO_FORNECEDOR_INGREDIENTE
(cd_pedido_fornecedor_ingrediente,
cd_pedido_fornecedor, qt_pedido_fornecedor_ingrediente,
vl_unitario_pedido_fornecedor_ingrediente, id_ingrediente)
VALUES (4, 4, 12.0, 6.0, 4);

INSERT INTO PEDIDO_FORNECEDOR_INGREDIENTE
(cd_pedido_fornecedor_ingrediente,
cd_pedido_fornecedor, qt_pedido_fornecedor_ingrediente,
```

```
vl_unitario_pedido_fornecedor_ingrediente, id_ingrediente)
VALUES (5, 5, 20.0, 8.0, 5);

-- Ingrediente F3
CREATE TABLE INGREDIENTE_F3 (
id_ingrediente INT NOT NULL,
constraint INGREDIENTE_PK PRIMARY KEY (id_ingrediente)
);

-- inserir na tabela Ingrediente F3
INSERT INTO INGREDIENTE_F3 (id_ingrediente) VALUES (1);

INSERT INTO INGREDIENTE_F3 (id_ingrediente) VALUES (2);

INSERT INTO INGREDIENTE_F3 (id_ingrediente) VALUES (3);

INSERT INTO INGREDIENTE_F3 (id_ingrediente) VALUES (4);

INSERT INTO INGREDIENTE_F3 (id_ingrediente) VALUES (5);

-- criar uma consulta com junção de tabelas
SELECT pf.cd_pedido_fornecedor, pf.dt_pedido_fornecedor,
pf.vl_total_pedido_fornecedor,
pfi.cd_pedido_fornecedor_ingrediente,
pfi.qt_pedido_fornecedor_ingrediente,
pfi.vl_unitario_pedido_fornecedor_ingrediente,
i.nm_ingrediente, i.dt_validade_ingrediente
FROM PEDIDO_FORNECEDOR pf
JOIN PEDIDO_FORNECEDOR_INGREDIENTE pfi ON
pf.cd_pedido_fornecedor = pfi.cd_pedido_fornecedor
JOIN INGREDIENTE i ON pfi.id_ingrediente = i.cd_ingrediente;

-- criar uma consulta com função de grupo
SELECT f.nm_fornecedor, COUNT(pf.cd_pedido_fornecedor) AS
total_pedidos
FROM PEDIDO_FORNECEDOR pf
JOIN FORNECEDOR f ON pf.cd_fornecedor = f.cd_fornecedor
```

```
GROUP BY f.nm_fornecedor;

--
**********************************************************************
***************
--                                TAREFA 02
--
**********************************************************************
***************

-- 1) Crie uma consulta para exibir o sobrenome do
funcionário, sua matrícula e o nome do departamento que ele
está alocado.

SELECT e.LAST_NAME AS "Sobrenome",
       e.EMPLOYEE_ID AS "Matrícula",
       d.DEPARTMENT_NAME AS "Departamento"
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID;

-- 2) Crie uma lista única de todos os cargos existentes no
departamento 80. Inclua a localização deste departamento.

SELECT DISTINCT j.JOB_TITLE AS Cargo,
                l.CITY AS Cidade,
                l.STATE_PROVINCE AS Estado
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID
JOIN LOCATIONS l ON d.LOCATION_ID = l.LOCATION_ID
JOIN JOBS j ON e.JOB_ID = j.JOB_ID
WHERE d.DEPARTMENT_ID = 80;

-- 3) Crie uma consulta para exibir o sobrenome do
funcionário, o nome do departamento, a localização e a cidade
de todos os funcionários que recebem comissão.

SELECT e.LAST_NAME AS Sobrenome,
```

```
       d.DEPARTMENT_NAME AS Departamento,
       l.STREET_ADDRESS AS "Endereço",
       l.CITY AS Cidade
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID
JOIN LOCATIONS l ON d.LOCATION_ID = l.LOCATION_ID
WHERE e.COMMISSION_PCT IS NOT NULL;


-- 4) Exiba o sobrenome do funcionário e o nome do
departamento para todos os funcionários que possuem um a em
seus sobrenomes.

SELECT e.LAST_NAME AS Sobrenome,
       d.DEPARTMENT_NAME AS Departamento
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID
WHERE e.LAST_NAME LIKE '%a%';

-- 5) Crie uma consulta para exibir o sobrenome, o cargo, o
número e o nome do departamento para todos os funcionários que
trabalham em Toronto.

SELECT e.LAST_NAME AS Sobrenome,
       e.JOB_ID AS Cargo,
       e.EMPLOYEE_ID AS Numero,
       d.DEPARTMENT_NAME AS Departamento
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID
JOIN LOCATIONS l ON d.LOCATION_ID = l.LOCATION_ID
WHERE l.CITY = 'Toronto';

-- 6) Exiba o sobrenome e o número do funcionário junto com o
sobrenome e o número do gerente. Coloque um label ou apelido
nas colunas.

SELECT
```

```
    e1.LAST_NAME AS Sobrenome_Funcionario,
    e1.EMPLOYEE_ID AS Numero_Funcionario,
    e2.LAST_NAME AS Sobrenome_Gerente,
    e2.EMPLOYEE_ID AS Numero_Gerente
FROM
    EMPLOYEES e1
JOIN
    EMPLOYEES e2 ON e1.MANAGER_ID = e2.EMPLOYEE_ID;

-- 7) Exibir todos os funcionários incluindo King, que não tem
gerente.
-- obs: Exibir todos os funcionários que possuem "King" no
nome, que não tem gerente.
SELECT
    e1.LAST_NAME AS Sobrenome_Funcionario,
    e1.EMPLOYEE_ID AS Numero_Funcionario
FROM
    EMPLOYEES e1
LEFT JOIN
    EMPLOYEES e2 ON e1.MANAGER_ID = e2.EMPLOYEE_ID
WHERE
    e1.LAST_NAME LIKE '%King%' AND e1.MANAGER_ID IS NULL;

-- 8) Crie uma consulta que exibirá o sobrenome dos
funcionários, o número do departamento e todos os funcionários
que trabalham no mesmo departamento de um determinado
funcionário. Forneça a cada coluna um label apropriado.

-- a) The maximum result size reached.

SELECT
    e1.LAST_NAME AS Sobrenome_Funcionario,
    e1.DEPARTMENT_ID AS Numero_Departamento,
    e2.LAST_NAME AS Sobrenome_Colega,
    e2.EMPLOYEE_ID AS Numero_Colega
FROM
    EMPLOYEES e1
```

```sql
JOIN
    EMPLOYEES e2 ON e1.DEPARTMENT_ID = e2.DEPARTMENT_ID
WHERE
    e1.EMPLOYEE_ID <> e2.EMPLOYEE_ID;

-- 9) Crie uma consulta que exiba o sobrenome, o cargo, o nome do departamento, o salário e a faixa salarial. DICA: utilize a tabela JOBS.

SELECT
    e.LAST_NAME AS Sobrenome,
    j.JOB_TITLE AS Cargo,
    d.DEPARTMENT_NAME AS Departamento,
    e.SALARY AS Salario,
    j.MIN_SALARY || ' - ' || j.MAX_SALARY AS Faixa_Salarial
FROM
    EMPLOYEES e
JOIN
    JOBS j ON e.JOB_ID = j.JOB_ID
JOIN
    DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID;

-- 10) Crie uma consulta para exibir o sobrenome e a data de admissão de qualquer funcionário admitido após o funcionário Davies

SELECT
    LAST_NAME AS Sobrenome,
    HIRE_DATE AS Data_Admissao
FROM
    EMPLOYEES
WHERE
    HIRE_DATE > (SELECT HIRE_DATE FROM EMPLOYEES WHERE LAST_NAME = 'Davies');
```

```
--
**********************************************************************
***************
--                                TAREFA 03
--
**********************************************************************
***************

-- ESTUDO DE CASO: JOIN E FUNÇÃO DE GRUPO

-- 1.   Exibir o nome do proprietário e o nome do veiculo;
SELECT p.nm_proprietario, v.nm_veiculo
FROM proprietario p JOIN veiculo v
ON (p.cd_cpf_proprietario = v.cd_cpf_proprietario)

-- 2.   Exibir o CPF do proprietário, a marca do veiculo e a
data do licenciamento;

        SELECT p.cd_cpf_proprietario, v.nm_marca,
l.dt_licenciamento
    FROM PROPRIETARIO p, VEICULO v, LICENCIAMENTO l
    WHERE p.cd_cpf_proprietario = v.cd_cpf_proprietario AND
l.cd_placa_veiculo = v.cd_placa_veiculo

-- 3.   Exibir a cor do veículo, a placa do veiculo e o valor
do licenciamento;

    SELECT v.nm_cor, v.cd_placa_veiculo, l.vl_licenciamento
         FROM  veiculo v, licenciamento l
         WHERE l.cd_placa_veiculo = v.cd_placa_veiculo

-- 4.   Exibir o nome do proprietário e o valor do
licenciamento

    SELECT p.nm_proprietario, l.vl_licenciamento
         FROM  veiculo v, licenciamento l, proprietário p
        WHERE p.cd_cpf_proprietario = v.cd_cpf_proprietario
```

```
        AND l.cd_placa_veiculo = v.cd_placa_veiculo

-- 5.   Alterar o anterior para exibir somente quando o valor
estiver no intervalo de R$ 500,00 a R$ 600,00.

    SELECT p.nm_proprietario, l.vl_licenciamento
         FROM  veiculo v, licenciamento l, proprietário p
        WHERE p.cd_cpf_proprietario = v.cd_cpf_proprietario
        AND l.cd_placa_veiculo = v.cd_placa_veiculo
GROUP BY l.vl_licenciamento BETWEEN 500 AND 600

--
**************************************************************
***************
--                              TAREFA 04
--
**************************************************************
***************


-- 1.Crie uma consulta para exibir o sobrenome e a data de
admissão de todos os funcionários no mesmo departamento do
funcionário com sobrenome Zlotkey. Exclua Zlotkey. FILEIRA 1

SELECT last_name, hire_date
FROM employees
WHERE department_id = (
    SELECT department_id
    FROM employees
    WHERE last_name = 'Zlotkey'
)
AND last_name != 'Zlotkey';

-- 2.Cria uma consulta para exibir o número e o nome de todos
os funcionários que recebam mais que o salário médio.
Classifique os resultados, por salário, em ordem decrescente.
FILEIRA 3
```

```sql
SELECT employee_id, last_name
FROM employees
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
)
ORDER BY salary DESC;

-- 3.Exiba o sobrenome do funcionário, o número do
departamento e o cargo de todos os funcionários cuja
localização do departamento seja 1700. FILEIRA 2

SELECT e.last_name, e.department_id, e.job_id
FROM employees e
JOIN departments d ON e.department_id = d.department_id
WHERE d.location_id = 1700;

-- 4. Exiba o sobrenome e o salário dos funcionários que se
reportam a King. FILEIRA 4

SELECT e.last_name, e.salary
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id
WHERE m.last_name = 'King';

-- 5. Exibir o nome do departamento somente quando o menor
salario for maior que o do depto. 50. FILEIRA 5

SELECT department_name
FROM departments
WHERE (SELECT MIN(salary) FROM employees WHERE department_id =
departments.department_id) > (
    SELECT MAX(salary) FROM employees WHERE department_id = 50
);
```

```sql
-- 6.Exibir todos os funcionarios que nao tenham NENHUM
SUBORDINADO, utilizando subquerie (utilizar NOT IN) FILEIRA 3

SELECT employee_id, last_name
FROM employees
WHERE employee_id NOT IN (
    SELECT manager_id
    FROM employees
    WHERE manager_id IS NOT NULL
);

-- 7.Exibir id, nome, nome do departamento de quem tem o mesmo
cargo do empregado com id 167. FILEIRA 4

SELECT e.employee_id, e.last_name, e.job_id
FROM employees e
WHERE e.job_id = (
    SELECT job_id
    FROM employees
    WHERE employee_id = 167
);

-- 8. Com subquerie, exibir os deptos que nao tem funcionario.
FILEIRA 2

SELECT department_id, department_name
FROM departments
WHERE department_id NOT IN (
    SELECT department_id
    FROM employees
);

-- duvida (todos departamentos possui funcionarios)

SELECT department_id, COUNT(employee_id)
FROM employees
GROUP BY department_id;
```

```
--
**********************************************************************
***************
--                                TAREFA 05
--
**********************************************************************
***************

-- Criar uma view chamada vw_cargo que tenha as colunas job_id
e job_title;
CREATE OR REPLACE VIEW vw_cargo AS
SELECT job_id, job_title FROM jobs;

-- Exibir a estrutura da view
DESCRIBE vw_cargo;

-- Exibir os cargos que tenham um i minusculo em seu nome
SELECT * FROM vw_cargo WHERE LOWER(job_title) LIKE '%i%';

-- Utilizando o dicionário de dados, exibir a consulta que
criou a view
SELECT text FROM all_views WHERE view_name = 'VW_CARGO';

-- Inserir um cargo com id = 'PL' nome= 'Programador PL/SQL'
INSERT INTO jobs (job_id, job_title) VALUES ('PL',
'Programador PL/SQL');

-- Alterar a view para que seja somente de leitura
DROP VIEW vw_cargo;
CREATE OR REPLACE VIEW vw_cargo AS
SELECT job_id, job_title FROM jobs WHERE 1=0;

-- Tentativa de inserir um cargo com id = 'ES' nome =
'Engenheiro de Software'
-- Isso resultará em um erro porque a view é de apenas leitura
```

```sql
-- Excluir a view
DROP VIEW vw_cargo;

-- Criar um sinonimo chamado cargo para jobs
CREATE SYNONYM cargo FOR jobs;

-- Criar a tabela aluno com matricula INTEGER PRIMARY KEY e
nome VARCHAR(20)
CREATE TABLE aluno (
    matricula INTEGER PRIMARY KEY,
    nome VARCHAR(20)
);

-- Criar uma sequencia chamada seq_aluno que inicie em 10 com
incremento 10
CREATE SEQUENCE seq_aluno START WITH 10 INCREMENT BY 10;

-- Utilizando a sequencia criada, inserir 3 registros na
tabela ALUNO
INSERT INTO aluno (matricula, nome) VALUES (seq_aluno.nextval,
'Nome1');
INSERT INTO aluno (matricula, nome) VALUES (seq_aluno.nextval,
'Nome2');
INSERT INTO aluno (matricula, nome) VALUES (seq_aluno.nextval,
'Nome3');

--
**********************************************************
***************
--                              TAREFA 06
--
**********************************************************
***************

-- 1)Criar o tipo JURIDICAS com inscricao estadual varchar(30)
e cnpj char(14) que herda as caracteristicas de PESSOAS
```

```sql
CREATE TYPE juridica UNDER pessoas (inscricao_estadual
VARCHAR(30), cnpj CHAR(14));

-- Criar a tabela PES_JURIDICAS com base no tipo JURIDICAS

CREATE TABLE pes_juridicas OF juridica

desc pes_juridicas

-- acrescentar chave primaria na tabela na coluna cdpessoa:

alter table pes_juridicas add constraint pesjuridica_pk
primary key (cdpessoa)

-- Inserir registro na tabela PES_JURIDICA

insert into pes_juridicas values (1, 'Isadora', '16236617855',
'0123456789123')

-- Exibir os registros da tabela PES_JURIDICA

select * from pes_juridicas

--
**********************************************************
***************
--                              TAREFA 07
--
**********************************************************
***************

-- LABORATÓRIO DE BANCO DE DADOS - REVISÃO P1

-- 1) Exibir  o nome do cliente e o valor total das faturas.
Ordenar em ordem crescente pelo nome

SELECT nm_cliente, vl_total_fatura from CLIENTE c ,
```

```
    JOIN FATURA f ON c.cd_cliente = f.cd_cliente ORDER BY
c.nm_cliente ASC

-- 2) Exibir o nome dos produtos e a quantidade de itens da
fatura deste produto

SELECT p.nm_produto, SUM(i.qt_item_fatura) AS quantidade_itens
FROM produto p
JOIN item_fatura i ON p.cd_produto = i.cd_produto
GROUP BY p.nm_produto;



-- 3) Exibir o nome do cliente, a data da fatura e a média do
valor total da fatura.  Ordenar em ordem decrescente pela
média.
-- E NÃO ACEITA ORDENAR POR APELIDO DE COLUNA

SELECT
  cliente.nm_cliente AS NomeCliente,
  fatura.dt_fatura AS DataFatura,
  AVG(fatura.vl_total_fatura) AS MediaValorTotal
FROM
  cliente
JOIN item_fatura ON cliente.cd_cliente =
item_fatura.cd_cliente
JOIN fatura ON item_fatura.cd_fatura = fatura.cd_fatura
GROUP BY
  cliente.nm_cliente,
  fatura.dt_fatura
ORDER BY
  AVG(fatura.vl_total_fatura) DESC

-- 4) Alterar o exercício 3 para exibir somente quando a média
for igual ou maior que 3000. FALTA AGRUPAR

SELECT c.nm_cliente, f.dt_fatura, AVG(f.vl_total_fatura)
FROM cliente c JOIN fatura f
```

```
ON (c.cd_cliente = f.cd_cliente)
GROUP BY c.nm_cliente, f.dt_fatura
HAVING AVG(f.vl_total_fatura) >= 3000
ORDER BY AVG(f.vl_total_fatura) desc


-- 5) Exibir o código e o email dos clientes e a quantidade de
itens da fatura. Criar uma subconsulta dos clientes que
tiveram faturas emitidas na mesma data que o cliente com
código 171;
SELECT c.cd_cliente, c.nm_email, COUNT(f.cd_fatura)
FROM cliente c JOIN fatura f
ON(c.cd_cliente = f.cd_cliente)
WHERE c.dt_fatura in (SELECT dt_fatura FROM cliente WHERE
cd_fatura= 171)
GROUP BY c.cd_cliente, c.nm_cliente


-- 6) Criar uma subconsulta que exiba todos os produtos que
nunca foram vendidos.
SELECT cd_produto, nm_produto
FROM produto WHERE cd_produto
NOT IN ( SELECT cd_produto   FROM item_fatura );

-- 7) Utilizando os operadores de conjunto, exibir todos os
códigos de produtos que já foram vendidos;
SELECT DISTINCT p.cd_produto
FROM produto p
WHERE EXISTS (
    SELECT 1
    FROM item_fatura v
    WHERE v.cd_produto = p.cd_produto
);

-- 8) Alterar o exercício 7 para que a solução seja com
subconsulta;
```

```sql
SELECT DISTINCT p.cd_produto
FROM produto p
WHERE p.cd_produto IN (
SELECT DISTINCT cd_produto
FROM item_fatura
);

-- 9) Alterar o exercício 7 para que a solução seja com junção de tabelas;

SELECT p.cd_produto
FROM produto p
JOIN item_fatura i ON p.cd_produto = i.cd_produto;


-- 10) Utilizando os operadores de conjunto, exibir todos os clientes que nunca tiveram faturas emitidas;

SELECT DISTINCT c.nm_cliente
FROM cliente c
WHERE NOT EXISTS (
    SELECT 1
    FROM fatura v
    WHERE v.cd_cliente = c.cd_cliente
);

-- 11) Alterar o exercício 10 para que a solução seja com subconsulta;

SELECT * FROM CLIENTE
WHERE cd_cliente NOT IN (
    SELECT cd_cliente
    FROM FATURA);

-- 12) Alterar o exercício 10 para que a solução seja com junção de tabelas;
```

```sql
SELECT c.cd_cliente, c.nm_cliente
 FROM CLIENTE c LEFT JOIN FATURA f
ON c.cd_cliente = f.cd_cliente
WHERE f.cd_cliente IS NULL;

-- 13) Criar um índice chamado idx_cliente para a coluna
nm_cliente da tabela CLIENTE;

create index idx_cliente on CLIENTE ( nm_cliente ); (feito
pelo Afonso)

-- 14) Consulte os índices criados no usuário logado;
SELECT index_name FROM user_indexes;

-- 15) Excluir o índice chamado idx_cliente;
DROP index idx_cliente

-- 16) Criar uma sequência chamada seq_produto que inicie em
100 com incremento de 3, sem valores na memória cachê;
CREATE SEQUENCE seq_produto
    START WITH 100
    INCREMENT BY 3 NOCACHE

-- 17) Consultar informações das sequências existentes neste
usuário;

SELECT sequence_name, min_value, max_value, increment_by,
last_number
FROM user_sequences;


-- 18) Criar uma visão chamada vw_vendas que exiba o nome do
cliente, a data de fatura e a quantidade de itens que ele
adquiriu PARA QUE SEJA SOMENTE DE LEITURA.
CREATE VIEW vw_vendas (nome, data, quantidade)
    AS SELECT c.nm_cliente, f.dt_fatura, SUM(i.qt_item_fatura)
    FROM fatura f
```

```
    JOIN cliente c ON c.cd_cliente = f.cd_cliente
    JOIN item_fatura i ON i.cd_fatura = f.cd_fatura
    GROUP BY c.nm_cliente, f.dt_fatura
 WITH READ ONLY

-- 19) Criar uma visão chamada vw_vendas que exiba o nome do
cliente, a data de fatura e a quantidade de itens que ele
adquiriu;
CREATE VIEW vw_vendas (nome, data, quantidade)
    AS SELECT c.nm_cliente, f.dt_fatura, SUM(i.qt_item_fatura)
    FROM fatura f
    JOIN cliente c ON c.cd_cliente = f.cd_cliente
    JOIN item_fatura i ON i.cd_fatura = f.cd_fatura
    GROUP BY c.nm_cliente, f.dt_fatura


-- 20) Consultar o script da view criada usando o dicionário
de dados;
SELECT * FROM USER_VIEWS WHERE VIEW_NAME = 'VW_VENDAS';

-- 21) Criar um sinônimo chamado MERCADORIA para a tabela
PRODUTO;
CREATE SYNONYM MERCADORIA FOR PRODUTO

-- 22) Excluir a view criada e excluir o sinônimo criado.
DROP VIEW vw_vendas
DROP SYNONYM MERCADORIA

-- 23) Criar um tipo de objeto CONTA com código e data que
será super classe;
CREATE TYPE conta AS OBJECT (cdconta INTEGER , dtconta
DATETIME) NOT FINAL;

-- 24) Criar um tipo chamado CONTA_INVESTIMENTO que tenha o
saldo e herda as características do tipo CONTA;
CREATE TYPE conta_investimento UNDER conta (vlsaldo
DECIMAL(10, 2));
```

```sql
-- 25) Criar a tabela TB_INVESTIMENTO com base no tipo
CONTA_INVESTIMENTO (EMILY)
CREATE TABLE TB_INVESTIMENTO OF CONTA_INVESTIMENTO

-- 26) Inserir um registro na tabela TB_INVESTIMENTO
INSERT INTO tb_investimento VALUES (001, '12/12/2023',
10000,00);

-- 27) Consultar o registro inserido.
SELECT * FROM TB_INVESTIMENTO

-- 28) Criar o tipo POUPANCA com data_aniversario e saldo que
herda características do tipo CONTA.

CREATE TYPE poupanca UNDER CONTA
(data_aniversario date ,
saldo DECIMAL (10,2 ));

-- 29) Criar a tabela TB_POUPANCA com base no tipo POUPANCA
CREATE TABLE tb_poupanca OF poupanca
```