CONSULTAS LEITURA COM INSTRUÇÕES SQL SELECT

1. Encontre todos os nomes dos clientes que iniciam com 'Antonio'.

```
SELECT U.nome
FROM usuario as U, cliente as C
WHERE U.nome LIKE 'Antonio%' and U.id = C.usuario_id
```

2. Quais os nomes e telefones (DDD e número) dos clientes com nomes que terminam com 'Cooper'?

```
SELECT U.nome, T.ddd, T.numero
FROM usuario as U, usuario_telefone as T, cliente as C
WHERE U.nome LIKE '%Cooper' and U.id = T.usuario_id and U.id = C.usuario_id
```

3. Quais os nomes dos usuários registrados na tabela de usuario e que não são clientes? Podem ser retornados um ou mais nomes de usuários.

```
(SELECT nome
FROM usuario)
EXCEPT
(SELECT nome
FROM usuario, cliente
WHERE usuario.id = cliente.usuario_id)
```

4. Quais as descrições e preços dos produtos que não estão disponíveis na filial cujo identificador é 1?

```
SELECT P.preco, P.descricao
FROM produto as P, estoque as E
WHERE P.id = E.produto_id and E.filial_id = 1 and E.quantidade = 0
```

5. Quais os nomes dos funcionários que começam com 'Julie' e que possuem telefones com DDD 81? A consulta deverá retornar, também, o DDD e número do telefone do funcionário.

```
SELECT U.nome, T.ddd, T.numero
FROM usuario as U, funcionario as F, usuario_telefone as T
WHERE U.id = F.usuario_id and U.id = T.usuario_id and U.nome LIKE 'Julie%' and T.ddd =
81
```

6. Quais os identificadores e descrições dos produtos que estão faltando em estoque na filial de razão social 'THOUSAND OAKS'?

```
SELECT P.id, P.descricao
FROM filial as F, produto as P, estoque as E
WHERE P.id = E.produto_id and F.razao_social = 'THOUSAND OAKS' and F.id = E.filial_id
and E.quantidade = 0
```

7. Qual o preço do produto mais caro que faz parte da categoria de descrição 'RAM'?

```
SELECT max(p.preco) AS preco_maior_ram
FROM produto AS P, produto_categoria AS PC, categoria AS C
WHERE P.id = PC.produto_id AND C.id = PC.categoria_id AND c.descricao = 'RAM'
```

8. Elabore uma consulta que mostre as descrições e preços dos produtos que possuem mais de uma categoria associada.

```
SELECT P.descricao, P.preco, count(PC.categoria_id) AS total_categorias
FROM produto AS P, produto_categoria AS PC, categoria AS C
WHERE P.id = PC.produto_id AND C.id = PC.categoria_id
GROUP BY P.id, P.descricao, P.preco
HAVING count(P.id) > 1
```

9. Elabore uma consulta que mostre o valor total vendido por razão social de cada filial. A consulta deve levar em consideração todas vendas registradas na tabela venda.

```
SELECT F.razao_social, SUM(P.preco * VI.quantidade)
FROM venda AS V, venda_item AS VI, produto AS P, filial AS F
WHERE V.id = VI.venda_id AND P.id = VI.produto_id AND F.id = V.filial_id
GROUP BY F.razao_social
```

10. Quantos clientes compraram uma quantidade total de unidades de produtos acima da média da quantidade total de unidades compradas por todos os clientes? A consulta deve levar em consideração todas vendas registradas na tabela venda.

```
WITH cliente_total AS (
SELECT v.cliente_id, sum(VI.quantidade) AS total_itens
   FROM venda AS V, venda_item AS VI, produto AS P
   WHERE V.id = VI.venda_id AND P.id = VI.produto_id
   GROUP BY V.cliente_id
)

SELECT count(CT.cliente_id)
FROM cliente_total AS CT
WHERE CT.total_itens > (SELECT avg(CT.total_itens) FROM cliente_total AS CT)
```

CONSULTAS DE ESCRITA COM INSTRUÇÕES SQL INSERT

1. Insira um novo cliente com nome 'Antonio José da Silva', endereço 'Rua X, 123', e-mail 'ajsilva@provedor.com', login 'ajsilva' e senha 'ajs123'. Observe que para inserir um cliente, um usuário correspondente a esse cliente deve ser criado primeiro. Após a inserção do novo usuário, verifique o id que foi atribuído e vincule esse id ao cliente 'Antonio José da Silva'. Suponha que não existam dois usuários com o mesmo nome.

```
WITH novo_usuario AS (
        INSERT INTO usuario (nome, endereco, email, login, senha)
        VALUES ('Antonio José da Silva', 'Rua X, 123', 'ajsilva@provedor.com', 'ajsilva',
'ajs123')
        RETURNING id
)

INSERT INTO cliente (usuario_id)
SELECT id FROM novo_usuario;
```

2. Insira um novo funcionário com nome 'Rafael João da Costa', endereço 'Rua Y, 456', e-mail 'rjcosta@provedor.com', login 'rjcosta', senha 'rjc456' e salário 2500. Observe que para inserir um funcionário, um usuário correspondente a esse funcionário deve ser criado primeiro. Após a inserção do novo usuário, verifique o id que foi atribuído e vincule esse id ao funcionário 'Rafael João da Costa'. Suponha que não existam dois usuários com o mesmo nome.

```
WITH novo_usuario AS (
        INSERT INTO usuario (nome, endereco, email, login, senha)
        VALUES ('Rafael João da Costa', 'Rua Y, 456', 'rjcosta@provedor.com', 'rjcosta',
'rjc456')
        RETURNING id
)

INSERT INTO funcionario (usuario_id, salario)
VALUES((SELECT id FROM novo_usuario), 2500);
```

3. Insira três telefones para o cliente 'Antonio José da Silva'. Os telefones são: DDD 85, número 98765432; DDD 85, número 99754208; e DDD 85, número 98639121. Suponha que não existam dois usuários com o mesmo nome.

```
INSERT INTO usuario_telefone (usuario_id, ddd, numero)
VALUES
((SELECT id FROM usuario WHERE nome = 'Antonio José da Silva'), 85, 98765432),
((SELECT id FROM usuario WHERE nome = 'Antonio José da Silva'), 85, 99754208),
((SELECT id FROM usuario WHERE nome = 'Antonio José da Silva'), 85, 98639121)
```

4. Insira dois telefones para o funcionário 'Rafael João da Costa'. Os telefones são: DDD 85, número 91290507 e DDD 81, número 90871001. Suponha que não existam dois usuários com o mesmo nome.

```
INSERT INTO usuario_telefone (usuario_id, ddd, numero)
VALUES
((SELECT id FROM usuario WHERE nome = 'Rafael João da Costa'), 85, 91290507 ),
((SELECT id FROM usuario WHERE nome = 'Rafael João da Costa'), 81, 90871001)
```

5. Insira um novo produto de descrição 'Produto X' e preço 23.5. Após a inserção, verifique o id que foi atribuído ao novo produto 'Produto X'. Suponha que não existam dois produtos com a mesma descrição. Associe o produto de descrição 'Produto X' a duas categorias existentes.

```
WITH novo_produto AS (
        INSERT INTO produto (descricao, preco)
        VALUES ('Produto X', 23.5)
        RETURNING id
)

INSERT INTO produto_categoria (produto_id, categoria_id)
VALUES ((SELECT id FROM novo_produto), 1), ((SELECT id FROM novo_produto), 2)
```

CONSULTAS DE ESCRITA COM INSTRUÇÕES SQL UPDATE

1. Atualize, aumentando em 10 unidades na tabela estoque, a quantidade dos produtos que foram vendidos para o cliente de id igual a 226.

```
UPDATE estoque
SET quantidade = quantidade + 10
WHERE produto_id IN (
   SELECT vi.produto_id
   FROM venda_item AS vi, venda AS v
   WHERE vi.venda_id = v.id
     AND v.cliente_id = 226
);
```

2. Atualize, aumentando em uma unidade, a quantidade de cada produto existente no estoque da filial de id igual a 1.

```
UPDATE estoque
SET quantidade = quantidade + 1
WHERE filial_id = 1;
```

3. Atualize, aumentando em 10%, o salário de cada funcionário que realizou pelo menos uma venda.

```
UPDATE funcionario
SET salario = salario * 1.1
WHERE usuario_id IN (
        SELECT DISTINCT funcionario_id
        FROM venda
)
```

CONSULTAS DE ESCRITA COM INSTRUÇÕES SQL DELETE

1. Remova todos os itens de vendas do cliente de nome 'Antonio José da Silva'. Após isso, remova todas as vendas que foram feitas para o cliente de nome 'Antonio José da Silva'. Suponha que não existem dois usuários com o mesmo nome.

```
DELETE FROM venda_item
WHERE venda_id IN (
   SELECT v.id
   FROM venda v, usuario u
   WHERE v.cliente_id = u.id
     AND u.nome = 'Antonio José da Silva'
);

DELETE FROM venda
WHERE cliente_id IN (
   SELECT id
   FROM usuario
   WHERE nome = 'Antonio José da Silva'
);
```

2. Remova todos os itens de vendas onde os produtos estão vinculados a mais de uma categoria.

```
DELETE FROM venda_item
WHERE produto_id IN (
   SELECT produto_id
   FROM produto_categoria
   GROUP BY produto_id
   HAVING COUNT(*) > 1
);
```