EXERCÍCIOS PRÁTICOS

Banco de Dados

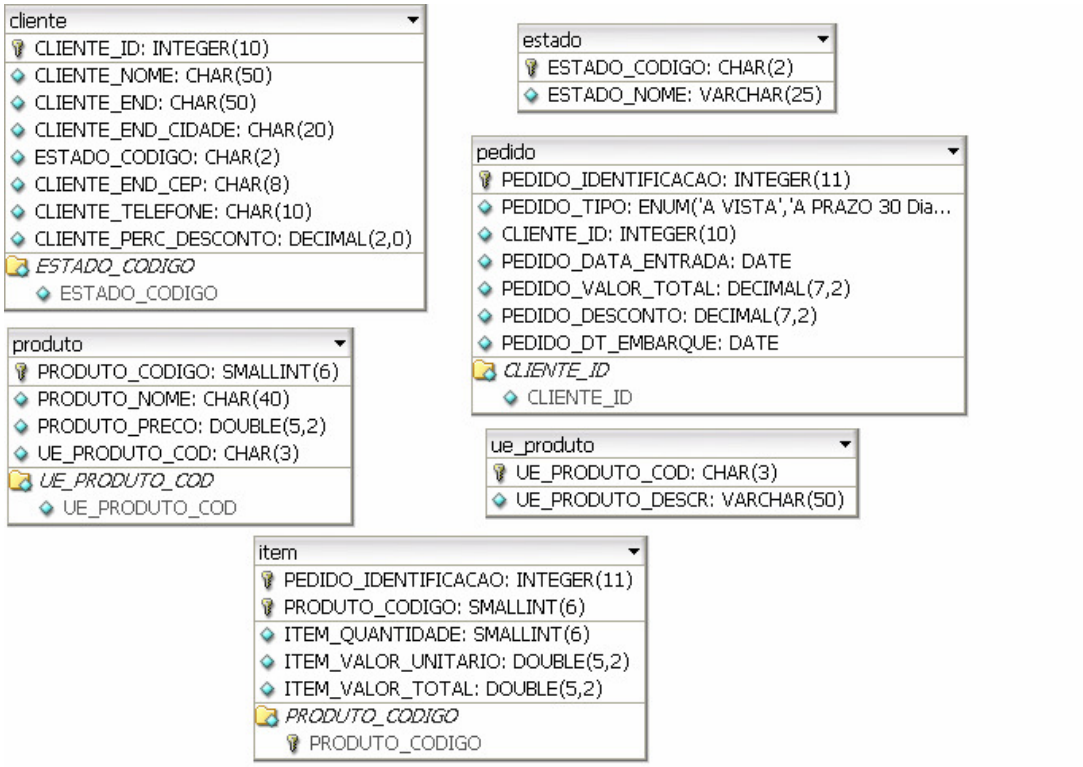
Introdução ao uso do SQL – *Structured Query Language*, considerando tanto DDL – *Data Definition Language* quanto DML – *Data Manipulation Language*.

Banco de Dados selecionado: MySQL

Para a execução dos comandos necessários pode-se utilizar um arquivo com extensão sql. Os tipos de comandos que podem ser disponibilizados em tal arquivo estão identificados nos exemplos abaixo.

1.1. Criação de um Banco de Dados

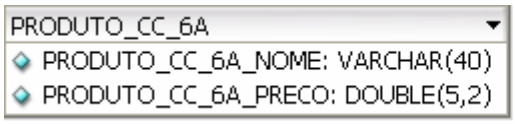
1.2. Exemplo

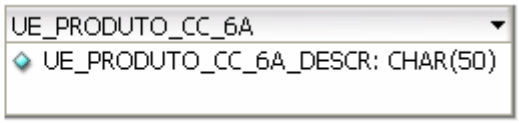


1.2. Aplicação de comando SQL-DDL

Considerando a sintaxe descrita para as operações SQL na parte 2 deste documento, para as tabelas

mostradas abaixo realize, sobre o BD criado, seguintes atividades:





1) Criar as tabelas acima usando o comando CREATE TABLE (criação de uma nova tabela). Entretanto, não defina ainda as colunas relacionadas com as PK (*Primary Key* – identificador único de cada registro da tabela) e FK (*Foreign Key* – importada de outra tabela, para relacionar tabelas) em ambas tabelas.

```
**************************
CREATE TABLE PRODUTO_CC_6A
(   produto_cc_6A_nome CHAR(40),
    produto_cc_6A_preco DOUBLE(5,2)  )

ou


CREATE TABLE PRODUTO_CC_6A
(      produto_cc_6A_codigo VARCHAR(2) NOT NULL,
       produto_cc_6A_nome CHAR(40),
       produto_cc_6A_preco DECIMAL(5,2),
       PRIMARY KEY (produto_cc_6A_codigo)  )

**************************
CREATE TABLE UE_PRODUTO_CC_6A
(  ue_produto_cc_6A_descr CHAR(50)  )

Ou


CREATE TABLE UE_PRODUTO_CC_6A
(   ue_produto_cc_6A_codigo VARCHAR(3) NOT NULL,
    ue_produto_cc_6A_descr CHAR(50)  )
**************************
```

2) Criar a chave primária para as duas tabelas através do comando ALTER TABLE (alteração de tabela). Inserir cada uma dessas PK na primeira posição de cada tabela.
   a. PK de UE_PRODUTO_CC_6A = ue_produto_cc_6A_codigo CHAR(3) NOT NULL
   b. PK de PRODUTO_CC_6A = produto_cc_6A_codigo SMALLINT NOT NULL

```
**************************
ALTER TABLE UE_PRODUTO_CC_6A
      ADD COLUMN ue_produto_cc_6A_codigo CHAR(3) NOT NULL FIRST;

ALTER TABLE UE_PRODUTO_CC_6A
      ADD PRIMARY KEY (ue_produto_cc_6A_codigo);

**************************
ALTER TABLE PRODUTO_CC_6A
      ADD COLUMN produto_cc_6A_codigo SMALLINT NOT NULL FIRST;

ALTER TABLE PRODUTO_CC_6A
      ADD PRIMARY KEY (produto_cc_6A_codigo);
**************************
```

3) Criar a chave estrangeira da tabela PRODUTO_CC_6A que aponta para a tabela UE_PRODUTO_cc_6A, analisando a ligação demonstrada no modelo com Integridade Referencial CASCADE.

   c. A FK de PRODUTO_CC_6A = ue_produto_cc_6A_codigo CHAR(3) NOT NULL;

   d. Essa chave aponta para o campo ue_produto_cc_6A_codigo da tabela UE_PRODUTO_ CC_6A

   A FK deve ser inserida na última posição do tabela PRODUTO_cc_6a.

```
**************************
ALTER TABLE PRODUTO_CC_6A
      ADD COLUMN
      ue_produto_cc_6A_codigo CHAR(3) NOT NULL AFTER produto_cc_6A_preco;

ALTER TABLE PRODUTO_CC_6a
      ADD FOREIGN KEY (ue_produto_cc_6A_codigo)
      REFERENCES UE_PRODUTO_cc_6A(ue_produto_cc_6A_codigo)
      ON UPDATE CASCADE;
**************************
```

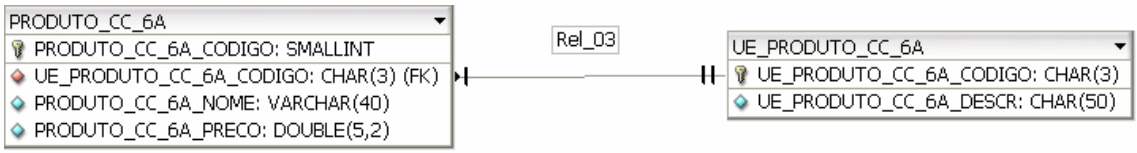
4) Criar um Índice único na tabela PRODUTO_CC_6A para a coluna produto_nome_CC_6a

```
**************************
CREATE UNIQUE INDEX indice_produto_nome
      ON  PRODUTO_CC_6A (produto_cc_6A_nome(5));
**************************
```

Para alterar uma coluna de uma tabela:

```
**************************
ALTER TABLE ue_produto_cc_6a
CHANGE COLUMN ue_produto_cc_6A_codigo
                  ue_produto_cc_6A_codigo CHAR(3) NOT NULL;
```

OBS: A execução das tarefas 2 e 3 sobre as tabelas acima deve ser a modificação das mesmas para a seguinte estrutura:



1.3. Aplicação de comando SQL-DML

Observação:

Antes e depois de executar qualquer comando de modificação (INSERT, UPDATE ou DELETE), verifique o conteúdo da tabela para se certificar de que a modificação tenha sido executada com sucesso.

Para os comandos INSERT, UPDATE E DELETE, utilizaremos as tabelas de apoio criadas acima (PRODUTO_CC_6A e EU_PRODUTO_CC_6A). Para o comando SELECT, utilizaremos as seguintes tabelas: CLIENTE, ESTADO, ITEM, PEDIDO, PRODUTO, UE_PRODUTO, conforme modelo de

dados definido pelo modelo apresentado em primeiro neste documento. Essas tabelas devem ser populadas com alguns dados de teste.

1.3.1. INSERT (inserção de um novo registro em uma tabela)

Executar inicialmente o seguinte comando SQL:

```
**************************
INSERT INTO UE_PRODUTO_CC_6A (ue_produto_cc_6a_codigo, ue_produto_cc_6a_descr)
SELECT ue_produto_cod, ue_produto_descr
FROM UE_PRODUTO;
**************************
```

Este comando faz uma cópia de cada registro da tabela UE_PRODUTO e o insere na tabela UE_PRODUTO_CC_6A.

Para os exercícios abaixo, utilize o comando INSERT do SQL, o qual tem a sintaxe geral mostrada abaixo na forma de um exemplo:

```
**************************
INSERT INTO <nome da tabela> (<colunas nas quais os valores serão inseridos separadas por vírgulas>)
VALUES (<valores para cada coluna listada);
**************************
```

Exemplo:

```
**************************
INSERT INTO PRODUTO_CC_6A (produto_cc_6a_codigo, ue_produto_cc_6a_codigo)
VALUES (112, 'MI');
**************************
```

Agora, ao invés de copiar os valores de uma outra tabela, eles foram informados diretamente.

1. Inserir uma linha na tabela PRODUTO_CC_6A informando todas as colunas, conforme formato geral.

```
**************************
INSERT INTO PRODUTO
VALUES (145,'Farinha',13.00,'Kg');
**************************
```

2. Inserir uma linha na tabela PRODUTO_CC_6A informando somente as colunas obrigatórias (aquelas que não podem ser vazias).

```
**************************
INSERT INTO PRODUTO_CC_6A (produto_cc_6a_codigo, ue_produto_cc_6a_codigo)
VALUES (112, 'MI');
**************************
```

3. Limpar toda a tabela PRODUTO_CC_6ª (isto é, excluir todos os seus registros), para poder executar o comando do item 4.

```
**************************
DELETE FROM PRODUTO_CC_6A;
**************************
```

4. Popular toda a tabela PRODUTO_CC_6A a partir de um comando SELECT (seleção) na tabela PRODUTO.

```
**************************
INSERT INTO PRODUTO_CC_6a (produto_cc_6a_nome, produto_cc_6a_preco,
produto_cc_6a_codigo, ue_produto_cc_6a_codigo)
SELECT produto_nome, produto_preco, produto_codigo, ue_produto_cod
FROM produto;
**************************
```

1.3.2. UPDATE (atualização)

Este comando altera os registros das tabelas do Banco de Dados. Sua sintaxe pode ser resumida com o seguinte exemplo:

```
**************************
UPDATE <nome da tabela a ser alterada>
SET <coluna a ser alterada> = <novo valor>
WHERE <condiçao de seleção do registro a ser alterado>;
**************************
```

Exemplo:

```
**************************
UPDATE produto_cc_6a
SET produto_cc_6a_preco = 12
WHERE produto_cc_6a_preco < 10;
**************************
```

Este exemplo atualiza a tabela PRODUTO_CC_6A, a coluna PRODUTO_CC_6A_PRECO com o valor 12, para aqueles registros onde o valor da coluna PRODUTO_CC_6A_PRECO sea menor do que 10.

1. Alterar a tabela PRODUTO_CC_6A, aplicando um aumento de 50% sobre o preço, somente para os produtos cujo preço for menor que 10.00.

```
**************************
UPDATE produto_cc_6a
SET produto_cc_6a_preco = produto_cc_6a_preco *1.5
WHERE produto_cc_6a_preco < 10;
**************************
```

2. Alterar a tabela PRODUTO_CC_6A, somente para o produto cujo código = 3, mudando o nome para “Feijão Preto e Branco”.

```
**************************
UPDATE produto_cc_6a
SET produto_cc_6a_nome = ‘Feijão Preto e Branco’
WHERE produto_cc_6a_codigo = 3;
**************************
```

3. Alterar o preço dos produtos na tabela PRODUTO_CC_6A dando um desconto de 40%, somente para os produtos cujo código da unidade de estoque (ue_produto_descr_CC_6A) seja = 5.

```
**************************
UPDATE produto_cc_6a
SET produto_cc_6a_preco = produto_cc_6a_preco *0.6
WHERE ue_produto_cc_6a_codigo = 5;
**************************
```

4. Alterar a descrição da unidade de estoque para “inutilizado” somente para as unidades de estoque para as quais não exista nenhum produto associado. (*****)

```
**************************
UPDATE ue_produto_cc_6a
SET ue_produto_cc_6a_descr = ‘inutilizado’
WHERE ue_produto_cc_6a_codigo NOT IN (SELECT DISTINCT
                                      ue_produto_cc_6a_codigo from produto_cc_6a);
**************************
```

Observar que neste caso a cláusula de condição questionou se o valor da coluna UE_PRODUTO_CC_6A_CODIGO não estava (NOT IN) numa seleção de valores da coluna UE_PRODUTO_CC_6A_CODIGO diferentes da tabela PRODUTO_CC_6A.

1.3.3. DELETE (deleção de registro)

1. Excluir da tabela PRODUTO_CC_6A o produto cujo código seja igual a 20.

```
**************************
DELETE FROM produto_cc_6a
WHERE ue_produto_cc_6a_codigo = 20;
**************************
```

2. Excluir na tabela PRODUTO_CC_6A somente os produtos cuja descrição da unidade de estoque (ue_produto_cc_6a_descr) seja = “litro”.

```
**************************
DELETE FROM produto_cc_6a
WHERE ue_produto_cc_6a_codigo IN (SELECT ue_produto_cc_6a_codigo FROM
ue_produto_cc_6a WHERE ue_produto_cc_6a_descr = 'LITRO');
**************************
```

1.3.4. SELECT (seleção de registro)

1. Selecionar todas as colunas (*) de todos os clientes da tabela CLIENTE.

```
**************************
SELECT *
FROM cliente;
**************************
```

2. Selecionar somente a coluna nome de todos os produtos.

```
**************************
SELECT produto_nome
FROM produto;
**************************
```

3. Selecionar as seguintes colunas dos clientes de maneira distinta: cidade, estado e cep .

```
**************************
SELECT DISTINCT cliente_end_cidade,estado_codigo, cliente_end_cep
FROM cliente;
**************************
```

4. Selecionar todos os pedidos para o cliente cujo código seja = 09 e cujo valor total seja maior que 50.

```
**************************
SELECT *
FROM pedido
WHERE cliente_id= 09 AND pedido_valor_total > 50.0;
**************************
```

5. Selecionar todos os pedidos cujo valor total seja menor que 100 ou maior que 500.

```
**************************
SELECT *
FROM pedido
WHERE pedido_valor_total < 100 OR pedido_valor_total > 500;
**************************
```

6. Selecionar todos os pedidos cuja data de entrada seja = 04/12/1999.

```
**************************
SELECT *
FROM pedido
WHERE pedido_data_entrada = '99/12/04'; ('1999/12/04' ou '99-12-04')
**************************
```

7. Selecionar todos os pedidos cuja data de entrada seja = 02/12/1999 e cujo valor total esteja entre 20 e 50.

```
**************************
SELECT *
FROM pedido
WHERE (pedido_valor_total BETWEEN 20 AND 50)
    AND pedido_data_entrada = '99/12/02';
**************************

**************************
SELECT *
FROM pedido
WHERE pedido_valor_total > 20 AND pedido_valor_total < 50
    AND pedido_data_entrada = '99/12/02';
**************************
```

8. Selecionar todos os clientes cujo código NÃO esteja entre 05 e 25.

```
**************************
SELECT *
FROM cliente
WHERE cliente_id NOT BETWEEN 5 AND 25;
**************************
```

ou

```
**************************
SELECT *
FROM cliente
WHERE cliente_id > 25 OR cliente_id < 5;
**************************
```

9. Selecionar todos os produtos cujo nome possua a primeira letra = 'P'.

```
**************************
SELECT *
FROM produto
WHERE produto_nome LIKE 'P%';
```

```
**************************
```

10. Selecionar todos os produtos cujo nome possua a string 'an' em qualquer posição do nome.

```
**************************
SELECT *
FROM produto
WHERE produto_nome LIKE '%an%';
**************************
```

11. Selecionar todos os produtos cujo nome comece com C ou F ou M, independente do resto.

```
**************************
SELECT *
FROM produto
WHERE produto_nome LIKE 'C%'
      OR produto_nome LIKE  'F%'
      OR produto_nome LIKE  'M%' ;
**************************
```

12. Selecionar todos os produtos cujo nome possua 8 caracteres, onde as duas primeiras letras sejam iguais a ‘ma’, independente do conteúdo das 6 próximas letras.

```
**************************
SELECT *
FROM produto
WHERE produto_nome LIKE 'ma______';
**************************

**************************
SELECT *
FROM produto
WHERE produto_nome LIKE ‘ma%’ AND LENGTH(produto_nome) = 8;
**************************
```

13. Selecionar todos os produtos cujo nome possua 7 caracteres e as duas primeiras letras não interessem, as 3 próximas sejam iguais a ‘ACA’ e as duas ultimas também não interessem.

```
**************************
SELECT *
FROM produto
WHERE produto_nome LIKE '__ACA__';
**************************
```

14. Selecionar todos os Produtos cujo Nome possua a string 'a_p' como parte do nome do produto, em qualquer posição do nome.

```
**************************
SELECT *
FROM produto
WHERE produto_nome LIKE '%a_p%';
**************************
```

15. Selecionar todos os clientes cuja UF seja ‘MG’ ou ‘ES’.

```
**************************
SELECT *
FROM cliente
```

```
WHERE estado_codigo = 'MG'
      OR estado_codigo = 'ES';
**************************
ou
**************************
SELECT *
FROM cliente
WHERE estado_codigo IN('MG','ES');
**************************
```

16. Selecionar todos os clientes cujo estado NAO seja RJ nem SP.

```
**************************
SELECT *
FROM cliente
WHERE estado_codigo <> 'RJ'
   AND estado_codigo <> 'SP';
**************************
Ou

**************************
SELECT *
FROM cliente
WHERE estado_codigo NOT IN ('RJ','SP');
**************************
```

17. Selecionar todos os produtos cujo preço seja menor que 20 e a unidade de estoque seja ‘kilograma’ ou ‘litro’.

```
**************************
SELECT *
FROM produto
WHERE produto_preco < 20
    AND ue_produto_cod IN (SELECT ue_produto_cod FROM ue_produto
                           WHERE ue_produto_descr = 'kilograma'
                                 OR ue_produto_descr = 'litro');
**************************
```

18. Selecionar todos os produtos cuja unidade de estoque seja ‘KG’ ou ‘UM’ e o preço seja maior que 10.

```
**************************
SELECT *
FROM produto
WHERE produto_preco > 10
    AND (ue_produto_codigo = 'KG' OR ue_produto_codigo = 'UM');
**************************
```

19. Selecionar todos os produtos cujo preço seja menor que 5, incluindo os produtos cujo preço esteja nulo.

```
**************************
SELECT *
FROM produto
WHERE produto_preco < 5 OR produto_preco is null;
**************************
```

20. Selecionar o nome e telefone dos clientes cujo código do cliente seja menor que 10 colocando um novo título (rótulo) para cada uma das colunas exibidas no resultado.

```
**************************
SELECT cliente_nome AS cliente, cliente_telefone AS telefoe
FROM cliente
WHERE cliente_id < 10;
**************************
```

21. Selecionar todos os produtos, demonstrando, o código e o nome do produto, o preço atual de cada produto, e o preço projetado com 30% de aumento.

```
**************************
SELECT produto_codigo AS codigo_do_produto, produto_nome AS nome_do_produto,
produto_preco AS preco_atual_do_produto,
produto_preco * 1.3 AS produto_com_aumento
FROM produto;
**************************
```

22. Selecionar todos os produtos por ordem de valor descendente, acrescentando um novo rótulo para as colunas exibidas.

```
**************************
SELECT produto_codigo AS codigo_do_produto, produto_nome AS nome_do_produto,
produto_preco AS preco_atual_do_produto,
ue_produto_cod AS unidade_medida_produto
FROM produto
ORDER BY produto_preco DESC;
**************************
```

23. Selecionar todos os Produtos por ordem de nome ascendente e valor descendente.

```
**************************
SELECT *
FROM produto
ORDER BY produto_nome ASC, produto_preco DESC;
**************************
```

24. Selecionar código do cliente, nome do cliente e data de entrada dos pedidos por ordem descendente de data de entrada, usando aliases (renomeação de tabela).

```
**************************
SELECT t1.cliente_id, t1.cliente_nome, t2.pedido_data_entrada
FROM cliente AS t1, pedido AS t2
WHERE t1.cliente_id = t2.cliente_id
ORDER BY t2.pedido_data_entrada DESC;
**************************
```

25. Contar a quantidade de pedidos feitos para o cliente 05.

```
**************************
SELECT COUNT(*) AS Total_de_pedidos
FROM pedido
WHERE cliente_id = 5;
**************************
```

26. Obter o MAIOR e o MENOR código de cliente da tabela cliente.

```
**************************
SELECT MAX(cliente_id) AS maior_id_cliente, MIN(cliente_id) AS menor_id_cliente
FROM cliente;
**************************
```

27. Obter o MAIOR e o MENOR valor de pedido.

```
**************************
SELECT MAX(pedido_valor_total) AS maior_valor_pedido,
MIN(pedido_valor_total) AS menor_valor_pedido
FROM pedido;
**************************
```

28. Obter o somatório do valor total geral de todos os pedidos.

```
**************************
SELECT SUM(pedido_valor_total) AS valor_total_pedidos
FROM pedido;
**************************
```

29. Obter o somatório do valor total das vendas da tabela pedido no período de 02/12/1999 até 04/12/1999.

```
**************************
SELECT SUM(pedido_valor_total) AS valor_total_pedidos
FROM pedido
WHERE pedido_data_entrada >= '99/12/02' AND pedido_data_entrada <= 9/12/04';
**************************
Ou

**************************
SELECT SUM(pedido_valor_total) AS valor_total_pedidos
FROM pedido
WHERE pedido_data_entrada BETWEEN '99/12/02' AND ‘9/12/04';
**************************
```

30. Obter a média do valor total das vendas da tabela pedidos do ano de 1999.

```
**************************
SELECT AVG(pedido_valor_total) AS valor_medio_pedidos
FROM pedido
WHERE pedido_data_entrada >= '99/01/01' AND pedido_data_entrada <= '99/12/31';

**************************
Ou

**************************
SELECT AVG (pedido_valor_total) Media
FROM pedido
WHERE EXTRACT(YEAR FROM pedido_data_entrada) = 1999
**************************

Ou
**************************
SELECT AVG(pedido_valor_total) AS valor_medio_pedidos
FROM pedido
WHERE pedido_data_entrada >= '99/01/01' AND pedido_data_entrada LIKE ‘99%’;
**************************
Ou

**************************
SELECT AVG (pedido_valor_total) Media
FROM pedido
WHERE YEAR(pedido_data_entrada) = 1999
**************************
```

31. Mostrar o código do produto e a média de quantidade por produto vendido.

```
**************************
SELECT produto_codigo, AVG(item_quantidade)
FROM item
GROUP BY produto_codigo;
**************************
```

32. Mostrar a quantidade de clientes por Unidade Federativa.

```
**************************
SELECT estado_codigo, COUNT(*)
AS Qtde_clientes
FROM cliente
GROUP BY estado_codigo;
**************************
```

33. Mostrar a quantidade de clientes por Unidade Federativa para clientes com desconto maior que 10%.

Ordem de avaliação:

1. Clausula WHERE
2. Clausula GROUP BY
3. Função de Agregação (SUM, MAX, MIN, COUNT)

```
**************************
SELECT estado_codigo, COUNT(*) AS Qtde_clientes
FROM cliente
WHERE cliente_perc_desconto > 10
GROUP BY estado_codigo;
**************************
```

34. Mostrar o código do pedido, e a média de valor dos mesmos, somente para os pedidos cujo valor está acima da média total.

Ordem de avaliação:

1. Clausula WHERE
2. Clausula GROUP BU
3. Função de Agregação (SUM, MAX, MIN, COUNT)
4. clausula HAVING

```
**************************
SELECT pedido_identificacao, AVG(pedido_valor_total) AS valor_medio
FROM pedido
GROUP BY pedido_identificacao
HAVING pedido_valor_total > valor_medio;
**************************
```

35. Mostrar código e o nome dos clientes que já fizeram pedidos, eliminando as repetições, classificando por nome do cliente descendente.

```
**************************
SELECT cliente_id, cliente_nome
FROM cliente
WHERE cliente_id IN (SELECT DISTINCT cliente_id FROM pedido)
```

```
ORDER BY cliente_nome DESC;
**************************
```

36. Mostrar código, nome dos produtos e valor total pedido por produto, dos produtos que já foram pedidos, classificando por nome do produto descendente selecionando somente os itens cuja soma total pedido por produto seja > 80, usando aliases.

```
**************************
SELECT SUM(t1.item_valor_total) AS ValorTotalPorProduto, t2.produto_nome AS nomeProduto, t2.produto_codigo AS produtoCodigo
FROM item AS t1, produto AS t2
WHERE t1.produto_codigo= t2.produto_codigo
GROUP BY produtoCodigo
HAVING ValorTotalPorProduto > 80
ORDER BY nomeProduto DESC;
**************************
```

37. Mostrar código do pedido, código e nome do cliente, código e nome do produto e descrição da unidade de estoque do produto.

```
**************************
SELECT t1.pedido_identificacao, t2.cliente_id, t3.cliente_nome, t4.produto_nome, t4.ue_produto_cod
FROM item AS t1, pedido AS t2, cliente AS t3, produto AS t4
WHERE t1.pedido_identificacao = t2. pedido_identificacao
    AND t2.cliente_id = t3.cliente_id
    AND t1.produto_codigo = t4.produto_codigo
**************************
```

38. Mostrar os dados dos produtos para os quais ainda não foi feito pedido.

```
**************************
SELECT produto_codigo, produto_nome
FROM produto
WHERE produto_codigo NOT IN (SELECT DISTINCT produto_codigo FROM item)
**************************
```

39. Mostrar código, nome e preço dos produtos cujo preço do produto seja maior que a média de preço de todos os produtos.

```
**************************
SELECT produto_codigo, produto_nome, produto_preco
FROM produto
WHERE produto_preco > (SELECT AVG(produto_preco) FROM produto);
**************************
```

40. Mostrar código, nome e preço dos produtos cujo preço seja maior que a média de preço de todos os produtos e a unidade de estoque do produto seja KG ou UM ou L e tenha a letra ‘h’ em qualquer parte do nome do produto.

```
**************************
SELECT produto_codigo, produto_nome, produto_preco, ue_produto_cod
FROM produto
WHERE (produto_preco > (SELECT AVG(produto_preco) FROM produto))
    AND (ue_produto_cod = 'KG' OR ue_produto_cod = 'UM' OR ue_produto_cod = 'L')
    AND (produto_nome LIKE '%h%');
**************************
```

Ou

```
**************************
SELECT produto_codigo, produto_nome, produto_preco, ue_produto_cod
FROM produto
WHERE (produto_preco > (SELECT AVG(produto_preco) FROM produto))
    AND ue_produto_cod IN ('KG','UM','L')
    AND produto_nome LIKE '%h%';
**************************
```

41. Mostrar os dados dos clientes que compraram no dia 02/12/1999 e que não compraram no dia 04/12/1999.

```
**************************
SELECT  cliente_nome, cliente_id
FROM cliente
WHERE cliente_id IN (SELECT DISTINCT cliente_id FROM pedido
                        WHERE pedido_data_entrada = '99/12/02')
      AND cliente_id NOT IN (SELECT DISTINCT cliente_id FROM pedido
                              WHERE pedido_data_entrada = '99/12/04');
**************************
```

42. Mostrar os dados dos produtos que nunca foram comprados por clientes que moram no estado do "Rio de Janeiro".

```
**************************
SELECT * from produto
WHERE produto_codigo NOT IN
                        (SELECT item.produto_codigo
                         FROM item, pedido, cliente
                         WHERE item.pedido_identificacao=pedido.pedido_identificacao
                              AND pedido.cliente_id = cliente.cliente_id
                              AND cliente.estado_codigo = 'RJ')
**************************
```

OBSERVAÇÕES:

Para disponibilizar um banco de dados para trabalhar com o Java:

- 1º. Passo: Instalar o MySQL (o servidor). Será solicitada uma senha para o root.
- 2º. Passo: instalar o mysql adm e o mysql query browser (http://dev.mysql.com/downloads/gui-tools/5.0.html) para poder criar bancos, alterar permissões, etc – São programas visuais para gerenciamento do BD;
  Ao executar será solicitado o estabelecimento de uma conexão. A primeira vez ela deve ser criada, a partir daí pode-se utilizar a mesma conexão, a não ser que se deseje criar uma nova;
  A partir daqui também é um bom ponto para criação dos próprios bancos de dados (ou schemas).
- 3º. Passo: Instalar driver JDBC para possibilitar uma conexão com mySQL - Disponibilizar o plug-in o Jconnector (http://dev.mysql.com/downloads/connector/). Deve estar no classpath para poder ser utilizado.
  A partir do Eclipse: Menu Window, item Preferences, Em Data Management, selecionar Driver Definitions, adicionar um driver MySQL, selecionando o JDBC apropriado (o arquivo jar)
- 4º. Para não ter problemas ao rodar o aplicativo o driver JDBC (arquivo jar) deve estar em no diretório de .../ jdk/jre/lib/ext.