UNIVERSIDADE POSITIVO

MARCIO HENRIQUE DE OLIVEIRA FRANCO - 38857570

BANCO DE DADOS

TRABALHO FINAL - A1

CURITIBA

2024

Parte 1: Modelagem do Banco de Dados

```
create table if not exists categorias (
  categoria_id int auto_increment primary key,
  nome varchar(50) not null,
  descricao text
);
```

A consulta define a criação de uma tabela chamada categorias caso ela ainda não exista. A tabela possui três colunas: categoria_id, nome e descricao. A coluna categoria_id é do tipo int e possui a propriedade auto_increment, ou seja, ela irá automaticamente atribuir um valor único e crescente para cada nova entrada, além de ser a chave primária da tabela. A coluna nome é do tipo varchar com um limite de 50 caracteres e não pode ser nula. A coluna descricao é do tipo text e pode conter descrições mais longas, sem uma restrição de tamanho.

```
create table if not exists fornecedores (
  fornecedor_id int auto_increment primary key,
  nome varchar(100) not null,
  contato varchar(50),
  endereco varchar(150),
  cidade varchar(50)
);
```

Essa consulta define a criação de uma tabela chamada fornecedores caso ela ainda não exista. A tabela possui cinco colunas: fornecedor_id, nome, contato, endereco e cidade. A coluna fornecedor_id é do tipo int e possui a propriedade auto_increment, atribuindo um valor único e crescente para cada novo registro, e é definida como a chave primária. A coluna nome é do tipo varchar com um limite de 100 caracteres e não pode ser nula. As colunas contato, endereco e cidade são do tipo varchar com limites de 50, 150 e 50 caracteres e permitem valores nulos.

```
create table if not exists produtos (
  produto_id int auto_increment primary key,
  nome varchar(100) not null,
  descricao text,
  preco decimal(10,2) not null,
  estoque int not null,
  categoria_id int,
  fornecedor_id int,
  foreign key (categoria_id) references categorias(categoria_id),
  foreign key (fornecedor_id) references fornecedores(fornecedor_id)
);
```

Essa consulta define a criação de uma tabela chamada produtos caso ela ainda não exista. A tabela possui oito colunas: produto_id, nome, descricao, preco, estoque, categoria_id e fornecedor_id. A coluna produto_id é do tipo int e possui a propriedade auto_increment, atribuindo um valor único e crescente para cada novo registro, e é definida como a chave primária. A coluna nome é do tipo varchar com um limite de 100 caracteres e não pode ser nula. A coluna descricao é do tipo text e permite descrições mais longas. As colunas preco e estoque são do tipo decimal e int e ambas possuem a restrição not null. As colunas categoria_id e fornecedor_id são do tipo int e representam chaves estrangeiras que referenciam as colunas categoria_id da tabela categorias e fornecedor_id da tabela fornecedores.

```
create table if not exists clientes (
  cliente_id int auto_increment primary key,
  nome varchar(100) not null,
  email varchar(100),
  telefone varchar(20),
  cpf varchar(14) not null,
  endereco varchar(150),
  cidade varchar(50)
);
```

A consulta define a criação de uma tabela chamada clientes caso ela ainda não exista. A tabela possui sete colunas: cliente_id, nome, email, telefone, cpf, endereco e cidade. A coluna cliente_id é do tipo int e possui a propriedade auto_increment, atribuindo um valor único e crescente para cada novo registro, e é definida como a chave primária. A coluna nome é do tipo varchar com um limite de 100 caracteres e não pode ser nula. As colunas email, telefone, endereco e cidade são do tipo varchar com limites de 100, 20, 150 e 50 caracteres e permitem valores nulos. A coluna cpf é do tipo varchar com limite de 14 caracteres e não pode ser nula.

```
create table if not exists funcionarios (
  funcionario_id int auto_increment primary key,
  nome varchar(100) not null,
  cargo varchar(50) not null,
  salario decimal(10,2) not null,
  data_contratacao date,
  email varchar(100),
  telefone varchar(20)
);
```

Essa consulta define a criação de uma tabela chamada funcionarios caso ela ainda não exista. A tabela possui sete colunas: funcionario_id, nome, cargo, salario, data_contratacao, email e telefone. A coluna funcionario_id é do tipo int e possui a propriedade auto_increment, atribuindo um valor único e crescente para cada novo registro, sendo definida como a chave primária. As colunas nome, cargo e salario são do tipo varchar e decimal com limites de 100, 50 e 10,2 caracteres e todas possuem a restrição not null. A coluna data_contratacao é da tipo date, email e telefone são varchar com limites de 100 e 20 caracteres e permitem valores nulos.

```
create table if not exists pedidos (
  pedido_id int auto_increment primary key,
  data_pedido date not null,
  cliente_id int,
  funcionario_id int,
  status varchar(20),
  foreign key (cliente_id) references clientes(cliente_id),
  foreign key (funcionario_id) references funcionarios(funcionario_id)
);
```

A consulta define a criação de uma tabela chamada pedidos caso ela ainda não exista. A tabela possui seis colunas: pedido_id, data_pedido, cliente_id, funcionario_id e status. A coluna pedido_id é do tipo int e possui a propriedade auto_increment, atribuindo um valor único e crescente para cada novo registro, sendo definida como a chave primária. A coluna data_pedido é do tipo date e não pode ser nula. As colunas cliente_id e funcionario_id são do tipo int e representam chaves estrangeiras que referenciam as colunas cliente_id da tabela clientes e funcionario_id da tabela funcionarios. A coluna status é do tipo varchar com um limite de 20 caracteres e permite valores nulos.

```
create table if not exists itens_pedido (
  item_pedido_id int auto_increment primary key,
  pedido_id int,
  produto_id int,
  quantidade int not null,
  preco_unitario decimal(10,2) not null,
  foreign key (pedido_id) references pedidos(pedido_id),
  foreign key (produto_id) references produtos(produto_id)
);
```

A consulta define a criação de uma tabela chamada itens_pedido caso ela ainda não exista. A tabela possui seis colunas: item_pedido_id, pedido_id, produto_id, quantidade, preco_unitario. A coluna item_pedido_id é do tipo int e

possui a propriedade auto_increment, atribuindo um valor único e crescente para cada novo registro, sendo definida como a chave primária. As colunas pedido_id e produto_id são do tipo int e representam chaves estrangeiras que referenciam as colunas pedido_id da tabela pedidos e produto_id da tabela produtos. A coluna quantidade é do tipo int e não pode ser nula, assim como a coluna preco_unitario, que é do tipo decimal com precisão de 10,2 caracteres.

```
create table if not exists pagamentos (
  pagamento_id int auto_increment primary key,
  pedido_id int,
  valor decimal(10,2) not null,
  metodo_pagamento varchar(50),
  data_pagamento date not null,
  foreign key (pedido_id) references pedidos(pedido_id)
);
```

Essa consulta define a criação de uma tabela chamada pagamentos caso ela ainda não exista. A tabela possui cinco colunas: pagamento_id, pedido_id, valor, metodo_pagamento e data_pagamento. A coluna pagamento_id é do tipo int e possui a propriedade auto_increment, atribuindo um valor único e crescente para cada novo registro, sendo definida como a chave primária. As colunas valor e data_pagamento são do tipo decimal e date e ambas possuem a restrição not null. A coluna pedido_id é do tipo int e representa uma chave estrangeira que referencia a coluna pedido_id da tabela pedidos. A coluna metodo_pagamento é do tipo varchar com um limite de 50 caracteres e permite valores nulos.

```
create table if not exists produtos_preco_original (
  produto_id int,
  preco decimal(10,2),
  foreign key (produto_id) references produtos(produto_id)
);
```

A consulta define a criação de uma tabela chamada produtos_preco_original caso ela ainda não exista. A tabela possui duas colunas: produto_id e preco. A coluna produto_id é do tipo int e representa uma chave estrangeira (foreign key) que referencia a coluna produto_id da tabela produtos. A coluna preco é do tipo decimal com precisão de 10,2 caracteres.

Parte 2: Operações de Manipulação de Dados

Parte 2.1: Inserção de Dados (INSERT)

```
insert into
  categorias (nome, descricao)
values
  ('Alimentos Básicos', 'Produtos essenciais para o dia a dia'),
  ('Bebidas', 'Variedades de bebidas para consumo'),
  ('Laticínios', 'Produtos derivados do leite'),
  ('Padaria', 'Itens de panificação frescos'),
  ('Limpeza e Higiene', 'Produtos para manutenção da limpeza e higiene');
```

A consulta insere cinco novos registros na tabela categorias. A instrução insert into é utilizada para adicionar novos dados nas colunas nome e descricao da tabela. Cada valor dentro dos parênteses representa um novo registro. O primeiro registro insere “Alimentos Básicos” na coluna nome e “Produtos essenciais para o dia a dia” na coluna descricao.

```
insert into
  fornecedores (nome, contato, endereco, cidade)
values
  ('Fornecedor de Alimentos Básicos', '41987654321', 'Rua A, 123', 'Curitiba'),
  ('Fornecedor de Bebidas', '41987654322', 'Avenida B, 456', 'Londrina'),
  ('Fornecedor de Laticínios', '41987654323', 'Praça C, 789', 'São Paulo'),
  ('Fornecedor de Padaria', '41987654324', 'Rua D, 101', 'Curitiba'),
  ('Fornecedor de Limpeza e Higiene', '41987654325', 'Avenida E, 202',
'Florianópolis');
```

Essa consulta insere cinco novos registros na tabela fornecedores. A instrução insert into é utilizada para adicionar novos dados nas colunas nome, contato, endereco e cidade da tabela. Cada valor entre parênteses representa um novo registro. O primeiro registro insere “Fornecedor de Alimentos Básicos” na

coluna nome, “41987654321” na coluna contato, “Rua A, 123” na coluna endereco e “Curitiba” na coluna cidade.

```
insert into
  produtos (nome, descricao, preco, estoque, categoria_id, fornecedor_id)
values
  ('Arroz', 'Arroz branco 1kg', 4.50, 25, 1, 1),
  ('Feijão', 'Feijão carioca 1kg', 6.80, 30, 1, 2),
  ('Macarrão', 'Macarrão espaguete 500g', 3.20, 15, 1, 3),
  ('Suco', 'Suco de uva 500ml', 8.50, 50, 2, 1),
  ('Refrigerante', 'Refrigerante 1L', 9.90, 10, 2, 3),
  ('Leite', 'Leite integral 1L', 3.80, 5, 3, 4),
  ('Manteiga', 'Manteiga com sal 200g', 7.50, 3, 3, 5),
  ('Queijo', 'Queijo mussarela 1kg', 30.00, 12, 3, 1),
  ('Pão', 'Pão de forma 500g', 5.50, 25, 4, 2),
  ('Bolo', 'Bolo de chocolate 1kg', 4.00, 20, 4, 2),
  ('Detergente', 'Detergente líquido 500ml', 2.50, 60, 5, 1),
  ('Sabão', 'Sabão em pó 1kg', 7.20, 40, 5, 2),
  ('Shampoo', 'Shampoo 350ml', 12.00, 7, 5, 3),
  ('Condicionador', 'Condicionador 350ml', 12.00, 2, 5, 4),
  ('Papel Higiênico', 'Papel higiênico 12 rolos', 15.00, 35, 5, 5);
```

Essa consulta insere 15 novos registros na tabela produtos. A instrução insert into é usada para adicionar dados nas colunas nome, descricao, preco, estoque, categoria_id e fornecedor_id. Cada valor entre parênteses representa um novo registro. O primeiro registro insere “Arroz” na coluna nome, “Arroz branco 1kg” na coluna descricao, “4.50” na coluna preco, “25” na coluna estoque, “1” na coluna categoria_id e “1” na coluna fornecedor_id.

```
insert into
  clientes (nome, email, telefone, cpf, endereco, cidade)
values
  ('Ana Silva', 'ana.silva@email.com', '41981234567', '123.456.789-00', 'Rua A,
100', 'Curitiba'),
  ('Bruno Souza', 'bruno.souza@email.com', '41981234568', '234.567.890-11',
'Avenida B, 200', 'São Paulo'),
  ('Carlos Lima', 'carlos.lima@email.com', '41981234569', '345.678.901-22',
'Praça C, 300', 'Rio de Janeiro'),
  ('Daniela Ferreira', 'daniela.ferreira@email.com', '41981234570',
'456.789.012-33', 'Rua D, 400', 'Belo Horizonte'),
  ('Eduardo Costa', 'eduardo.costa@email.com', '41981234571', '567.890.123-44',
'Avenida E, 500', 'Porto Alegre'),
  ('Fernanda Almeida', 'fernanda.almeida@email.com', '41981234572',
'678.901.234-55', 'Praça F, 600', 'Recife'),
  ('Gustavo Pereira', 'gustavo.pereira@email.com', '41981234573',
'789.012.345-66', 'Rua G, 700', 'Salvador'),
  ('Helena Martins', 'helena.martins@email.com', '41981234574',
'890.123.456-77', 'Avenida H, 800', 'Fortaleza'),
  ('Isabela Santos', 'isabela.santos@email.com', '41981234575', '901.234.567-88',
'Praça I, 900', 'Manaus'),
  ('João Oliveira', 'joao.oliveira@email.com', '41981234576', '012.345.678-99',
'Rua J, 1000', 'Curitiba');
```

A consulta insere dez novos registros na tabela clientes. A instrução insert into é utilizada para adicionar novos dados nas colunas nome, email, telefone, cpf, endereco e cidade. Cada valor entre parênteses representa um novo registro. O primeiro registro insere “Ana Silva” na coluna nome, “ana.silva@email.com” na coluna email, “41981234567” na coluna telefone, “123.456.789-00” na coluna cpf, “Rua A, 100” na coluna endereco e “Curitiba” na coluna cidade.

```
insert into
  funcionarios (nome, cargo, salario, data_contratacao, email, telefone)
values
  ('Mariana Oliveira', 'Gerente', 5000.00, '2022-01-15',
'mariana.oliveira@email.com', '41981234577'),
  ('Carlos Almeida', 'Vendedor', 2500.00, '2023-02-10',
'carlos.almeida@email.com', '41981234578'),
  ('Fernanda Sousa', 'Caixa', 2200.00, '2021-03-20', 'fernanda.sousa@email.com',
'41981234579'),
  ('Rodrigo Pereira', 'Estoque', 2000.00, '2020-04-25',
'rodrigo.pereira@email.com', '41981234580'),
  ('Ana Martins', 'Responsável por Pedidos', 2700.00, '2023-05-30',
'ana.martins@email.com', '41981234581'),
  ('João Silva', 'Responsável por Pedidos', 2700.00, '2022-06-12',
'joao.silva@email.com', '41981234582'),
  ('Maria Jose', 'Responsável por Pedidos', 2700.00, '2020-02-21',
'maria.jose@email.com', '41989237522');
```

A consulta insere sete novos registros na tabela funcionarios. A instrução insert into é utilizada para adicionar novos dados nas colunas nome, cargo, salario, data_contratacao, email e telefone. Cada valor entre parênteses representa um novo registro. O primeiro registro insere “Mariana Oliveira” na coluna nome, “Gerente” na coluna cargo, “5000.00” na coluna salario, “2022-01-15” na coluna data_contratacao, “mariana.oliveira@email.com” na coluna email e “41981234577” na coluna telefone.

```
insert into
  pedidos (data_pedido, cliente_id, funcionario_id, status)
values
  ('2024-11-01', 1, 2, 'Concluído'),
  ('2024-11-02', 2, 4, 'Em andamento'),
  ('2024-11-03', 3, 6, 'Concluído'),
  ('2024-11-04', 4, 6, 'Em andamento'),
  ('2024-11-05', 5, 5, 'Concluído'),
  ('2024-11-06', 6, 6, 'Em andamento'),
  ('2024-11-07', 7, 5, 'Concluído'),
  ('2024-11-08', 8, 6, 'Em andamento'),
  ('2024-11-09', 9, 5, 'Concluído'),
  ('2024-11-10', 10, 6, 'Em andamento');
```

A consulta insere dez novos registros na tabela pedidos. A instrução insert into é utilizada para adicionar novos dados nas colunas data_pedido, cliente_id, funcionario_id e status. Cada valor entre parênteses representa um novo registro. O primeiro registro insere “2024-11-01” na coluna data_pedido, “1” na coluna cliente_id, “2” na coluna funcionario_id e “Concluído” na coluna status.

```
insert into
  itens_pedido (pedido_id, produto_id, quantidade, preco_unitario)
values
  (1, 1, 100, 4.00),,
  (1, 4, 20, 8,00),
  (2, 2, 3, 6.80),
  (2, 5, 1, 4.00),
  (3, 3, 5, 3.20),
  (3, 7, 1, 3.80),
  (4, 6, 2, 9.90),
  (4, 8, 1, 7.50),
  (5, 9, 2, 30.00),
  (5, 10, 4, 5.50),
  (6, 11, 3, 2.50),
  (6, 12, 2, 7.20),
  (7, 13, 100, 12.00),
  (7, 14, 100, 12.00),
  (8, 15, 2, 15.00),
  (8, 1, 3, 4.50),
  (9, 2, 1, 6.80),
  (9, 3, 2, 3.20),
  (10, 4, 4, 8.50),
  (10, 5, 3, 4.00);
```

A consulta insere 20 registros na tabela itens_pedido. A instrução insert into é usada para adicionar dados nas colunas pedido_id, produto_id, quantidade e preco_unitario. Cada valor entre parênteses representa um novo registro. O primeiro registro insere “1” na coluna pedido_id, “1” na coluna produto_id, “2” na coluna quantidade e “4.50” na coluna preco_unitario.

```
insert into
  pagamentos (pedido_id, valor, metodo_pagamento, data_pagamento)
values
  (1, 50.00, 'Cartão de Crédito', '2024-11-01'),
  (2, 40.00, 'Boleto', '2024-11-02'),
  (3, 20.00, 'Transferência Bancária', '2024-11-03'),
  (4, 35.00, 'Cartão de Crédito', '2024-11-04'),
  (5, 30.00, 'Boleto', '2024-11-05'),
  (6, 45.00, 'Transferência Bancária', '2024-11-06'),
  (7, 24.00, 'Cartão de Crédito', '2024-11-07'),
  (8, 50.00, 'Boleto', '2024-11-08'),
  (9, 12.00, 'Transferência Bancária', '2024-11-09'),
  (10, 36.00, 'Cartão de Crédito', '2024-11-10');
```

Essa consulta insere dez novos registros na tabela pagamentos. A instrução insert into é utilizada para adicionar novos dados nas colunas pedido_id, valor, metodo_pagamento e data_pagamento. Cada valor entre parênteses representa um novo registro. O primeiro registro insere “1” na coluna pedido_id, “50.00” na coluna valor, “Cartão de Crédito” na coluna metodo_pagamento e “2024-11-01” na coluna data_pagamento.

Parte 2.2: Atualização de Dados (UPDATE)

```
insert into
  produtos_preco_original (produto_id, preco)
select
  produto_id, preco
from
  produtos
where
  categoria_id = 5;

update
  produtos
set
  preco = preco * 1.10
where
  produtos.categoria_id = 5;
```

A primeira parte insere novos registros na tabela produtos_preco_original, copiando o produto_id e o preco de todos os produtos que pertencem a categoria com categoria_id igual a 5. A instrução insert into...select é utilizada para copiar os dados da tabela produtos para a tabela produtos_preco_original, fazendo com que todos os produtos dessa categoria específica tenham seus preços originais armazenados.

A segunda parte atualiza os preços dos produtos na tabela produtos, aumentando o preço de cada produto da categoria com categoria_id igual a 5 em 10%. A instrução update...set é usada para modificar o valor da coluna preco, multiplicando o preço atual por 1.10, o que resulta em um aumento de 10%.

```
insert into
  produtos_preco_original (produto_id, preco)
select
  produto_id, preco
from
  produtos
where
  estoque > 20
and
  produto_id not in (select produto_id from produtos_preco_original);

update
  produtos
set
  preco = preco * 0.95
where
  produtos.estoque > 20;
```

A primeira parte insere novos registros na tabela produtos_preco_original, copiando o produto_id e o preco de todos os produtos que tem um estoque maior que 20 e que ainda não estão na tabela produtos_preco_original. A instrução insert into...select é utilizada para copiar os dados da tabela produtos para a tabela produtos_preco_original, fazendo com que apenas os produtos com estoque suficiente e que não foram anteriormente registrados sejam inseridos.

A segunda parte atualiza os preços dos produtos na tabela produtos, reduzindo o preço de cada produto com estoque superior a 20 em 5%. A instrução update...set é usada para modificar o valor da coluna preco, multiplicando o preço atual por 0.95, o que resulta em um desconto de 5%.

```
insert into
  produtos_preco_original (produto_id, preco)
select
  produto_id, preco
from
  produtos
where
  fornecedor_id = 2
and
  produto_id not in (select produto_id from produtos_preco_original);

update
  produtos
set
  preco = preco * 0.85
where
  produtos.fornecedor_id = 2;
```

A primeira parte insere novos registros na tabela produtos_preco_original, copiando o produto_id e o preco de todos os produtos fornecidos pelo fornecedor com fornecedor_id igual a 2 e que ainda não estão na tabela produtos_preco_original. A instrução insert into...select é utilizada para copiar os dados da tabela produtos para a tabela produtos_preco_original, fazendo com que apenas os produtos fornecidos pelo fornecedor especificado e que não foram anteriormente registrados sejam inseridos.

A segunda parte atualiza os preços dos produtos na tabela produtos, reduzindo o preço de cada produto fornecido pelo fornecedor com fornecedor_id igual a 2 em 15%. A instrução update...set é usada para modificar o valor da coluna preco, multiplicando o preço atual por 0.85, o que resulta em um desconto de 15%.

```
update
  clientes
set
  clientes.endereco = 'Rua Isadora Pinto, 230',
  clientes.telefone = '41988997070'
where
  cliente_id = 5;
```

A consulta atualiza os dados do cliente com cliente_id igual a 5 na tabela clientes. A instrução update...set é utilizada para modificar os valores das colunas endereco e telefone. O endereço é atualizado para “Rua Isadora Pinto, 230” e o telefone é atualizado para “41988997070”. O where faz com que essas mudanças afetem apenas o registro do cliente identificado pelo cliente_id 5.

```
update
  funcionarios
set
  cargo = 'Gerente',
  salario = salario * 1.20
where
  funcionario_id = 3;
```

Essa consulta atualiza os dados do funcionário com funcionario_id igual a 3 na tabela funcionarios. A instrução update...set é utilizada para modificar os valores das colunas cargo e salario. O cargo é atualizado para “Gerente” e o salário é aumentado em 20%, multiplicando o valor atual por 1.20. O where faz com que essas mudanças afetem apenas o registro do funcionário com id 3.

```
update
  pedidos
join
  pagamentos on pedidos.pedido_id = pagamentos.pedido_id
set
  status = 'Concluído'
where
  pagamentos.valor is not null;
```

A consulta realiza uma atualização na tabela pedidos, utilizando join com a tabela pagamentos. A instrução update...join...set é utilizada para modificar a coluna status na tabela pedidos. O status é atualizado para “Concluído” para todos os registros em que o valor de pagamentos.valor não é nulo. A junção é feita com base na coluna pedido_id, que está presente nas tabelas pedidos e pagamentos.

```
update
  produtos
join
  produtos_preco_original on produtos.produto_id =
produtos_preco_original.produto_id
set
  produtos.preco = produtos_preco_original.preco;
```

A consulta realiza uma atualização na tabela produtos utilizando join com a tabela produtos_preco_original. A instrução update...join...set é usada para modificar o valor da coluna preco na tabela produtos. O preço de cada produto na tabela produtos será atualizado para o preço original armazenado na tabela produtos_preco_original. A junção é feita com base na coluna produto_id, que deve estar presente nas duas tabelas.

```
update
  produtos
join
  itens_pedido on produtos.produto_id = itens_pedido.produto_id
set
  produtos.estoque = produtos.estoque - itens_pedido.quantidade;
```

Essa consulta atualiza a tabela produtos com base nas informações da tabela itens_pedido. A instrução update...join...set é usada para modificar o valor da coluna estoque na tabela produtos. O estoque de cada produto na tabela produtos será reduzido pela quantidade correspondente de itens vendidos, conforme registrado na tabela itens_pedido. A junção é feita com base na coluna produto_id, que deve estar nas duas tabelas.

Parte 3: Consultas SQL Avançadas

```
select
  produtos.nome, categorias.nome, fornecedores.nome
from
  produtos
inner join
  categorias on categorias.categoria_id = produtos.categoria_id
inner join
  fornecedores on fornecedores.fornecedor_id = produtos.fornecedor_id
where
  produtos.estoque > 10;
```

Essa consulta seleciona e retorna os nomes dos produtos, das categorias e dos fornecedores para todos os produtos que possuem mais de 10 unidades em estoque. O select especifica as colunas desejadas: produtos.nome, categorias.nome e fornecedores.nome. A partir da tabela produtos, a consulta realiza o inner join com as tabelas categorias e fornecedores para relacionar os produtos com as suas categorias e fornecedores, com base nos identificadores categoria_id e fornecedor_id. O where filtra os registros que apenas os produtos com mais de 10 unidades em estoque sejam mostrados no resultado.

```
select
  clientes.nome, pedidos.data_pedido, funcionarios.nome, pedidos.status
from
  pedidos
inner join
  clientes on clientes.cliente_id = pedidos.cliente_id
inner join
  funcionarios on funcionarios.funcionario_id = pedidos.funcionario_id
where
```

```
  clientes.cliente_id = 8 and
  pedidos.data_pedido >= curdate() - interval 30 day;
```

A consulta seleciona e retorna os nomes dos clientes, datas dos pedidos, nomes dos funcionários e status dos pedidos para todos os pedidos feitos pelo cliente com cliente_id igual a 8 nos últimos 30 dias. O select especifica as colunas desejadas: clientes.nome, pedidos.data_pedido, funcionarios.nome e pedidos.status. A partir da tabela pedidos, a consulta realiza inner join com as tabelas clientes e funcionarios para relacionar os pedidos aos seus clientes e funcionários, com base nos identificadores cliente_id e funcionario_id. O where filtra os registros para que apenas os pedidos feitos pelo cliente específico nos últimos 30 dias sejam colocados no resultado.

```
select
  produtos.nome, sum(itens_pedido.quantidade)
from
  produtos
inner join
  itens_pedido on produtos.produto_id = itens_pedido.produto_id
group by
  produtos.nome
having
  sum(itens_pedido.quantidade) > 5;
```

A consulta seleciona e retorna os nomes dos produtos e a soma das quantidades vendidas para aqueles produtos que tiveram mais de 5 unidades vendidas. A instrução select especifica as colunas desejadas: produtos.nome e a soma das quantidades (sum(itens_pedido.quantidade)). A partir da tabela produtos, a consulta realiza inner join com a tabela itens_pedido para relacionar os produtos com seus itens de pedido, com base no produto_id. O group by agrupa os resultados pelo nome do produto, e o having filtra os grupos para incluir apenas aqueles cuja soma das quantidades (sum(itens_pedido.quantidade)) é superior a 5.

```
select
  pagamentos.pedido_id,
  sum(pagamentos.valor),
  group_concat(pagamentos.metodo_pagamento SEPARATOR ', ')
from
  pagamentos
where
  pagamentos.data_pagamento >= curdate() - interval 60 day
group by
  pagamentos.pedido_id;
```

Essa consulta seleciona e retorna os IDs dos pedidos, a soma dos valores pagos e os métodos de pagamento concatenados para todos os pagamentos realizados nos últimos 60 dias. O select especifica as colunas desejadas: pagamentos.pedido_id, a soma dos valores (sum(pagamentos.valor)) e a concatenação dos métodos de pagamento (group_concat(pagamentos.metodo_pagamento SEPARATOR ', ')) com um separador de vírgula.

A partir da tabela pagamentos, o where filtra os registros para incluir apenas aqueles cuja data de pagamento é igual ou posterior a 60 dias a partir da data atual (curdate()). O group by agrupa os resultados pelo ID do pedido (pagamentos.pedido_id), permitindo que a soma dos valores e a concatenação dos métodos de pagamento sejam calculadas por pedido.

```
select
  pedidos.pedido_id,
  pedidos.data_pedido,
  pedidos.status,
  sum(pagamentos.valor)
from
  pedidos
inner join
  pagamentos on pedidos.pedido_id = pagamentos.pedido_id
where
  pagamentos.data_pagamento is null
  or pedidos.status = 'Em andamento'
group by
  pedidos.pedido_id
having
  sum(pagamentos.valor) > 100;
```

Essa consulta seleciona e retorna os IDs dos pedidos, datas dos pedidos, status dos pedidos e a soma dos valores pagos para todos os pedidos que atendem a duas condições: o pagamento não foi efetuado (pagamentos.data_pagamento is null) ou o status do pedido é “Em andamento”. O select especifica as colunas desejadas: pedidos.pedido_id, pedidos.data_pedido, pedidos.status e a soma dos valores (sum(pagamentos.valor)). A partir da tabela pedidos, a consulta realiza inner join com a tabela pagamentos para relacionar os pedidos aos seus pagamentos, com base no pedido_id. O where filtra os registros para incluir apenas os pedidos que atendem a uma das duas condições.

```
select
  produtos.nome,
  produtos.estoque,
  produtos.preco,
  fornecedores.nome
from
  produtos
inner join
  fornecedores on produtos.fornecedor_id = fornecedores.fornecedor_id
where
  produtos.estoque < 5
order by
  produtos.nome;
```

Essa consulta seleciona e retorna os nomes dos produtos, o estoque, o preço e os nomes dos fornecedores para todos os produtos que possuem menos de 5 unidades em estoque. O select especifica as colunas desejadas: produtos.nome, produtos.estoque, produtos.preco e fornecedores.nome. A partir da tabela produtos, a consulta realiza inner join com a tabela fornecedores para relacionar os produtos aos seus fornecedores, com base no fornecedor_id. O where filtra os registros para garantir que apenas os produtos com menos de 5 unidades em estoque sejam incluídos no resultado. O order by ordena os resultados pelo nome dos produtos (produtos.nome) em ordem alfabética.

```
select
  clientes.nome,
  sum(pagamentos.valor)
from
  clientes
inner join
  pedidos on clientes.cliente_id = pedidos.cliente_id
inner join
  pagamentos on pedidos.pedido_id = pagamentos.pedido_id
group by
  clientes.nome
having
  sum(pagamentos.valor) > 500;
```

A consulta seleciona e retorna os nomes dos clientes e a soma dos valores pagos por cada cliente que gastou mais de 500 reais. O select especifica as colunas desejadas: clientes.nome e a soma dos valores (sum(pagamentos.valor)). A partir da tabela clientes, a consulta realiza inner join com as tabelas pedidos e pagamentos para relacionar os clientes com seus pedidos e pagamentos, com base nos identificadores cliente_id e pedido_id. O group by agrupa os resultados pelo nome do cliente (clientes.nome), permitindo calcular a soma dos valores pagos por cliente. O having filtra esses grupos para incluir apenas aqueles cuja soma dos valores (sum(pagamentos.valor)) é superior a 500.

```
with total_vendas_por_funcionario as (
  select
    funcionarios.nome as nome_funcionario,
    sum(itens_pedido.quantidade * itens_pedido.preco_unitario) as total_vendas
  from
    funcionarios
  join
    pedidos on funcionarios.funcionario_id = pedidos.funcionario_id
  join
    itens_pedido on pedidos.pedido_id = itens_pedido.pedido_id
  group by
    funcionarios.nome
),
media_total_vendas as (
  select
    avg(total_vendas) as media_vendas
  from
    total_vendas_por_funcionario
)
select
  tvpf.nome_funcionario,
  tvpf.total_vendas
from
  total_vendas_por_funcionario tvpf
join
  media_total_vendas mtv on tvpf.total_vendas > mtv.media_vendas;
```

Essa consulta é organizada em duas 3 partes. A primeira parte, total_vendas_por_funcionario, calcula o total de vendas por funcionário, realizando joins entre as tabelas funcionarios, pedidos e itens_pedido e agrupando os resultados pelo nome dos funcionários. Isso permite somar o valor total das vendas por funcionário. A segunda parte, media_total_vendas, calcula a média das vendas totais por funcionário, utilizando a função avg() sobre os totais de vendas obtidos na primeira parte. A terceira parte seleciona os nomes dos funcionários e o total vendas

para aqueles que superam a média, juntando a total_vendas_por_funcionario com a media_total_vendas e aplicando a condição necessária.

```
select
  date_format(pedidos.data_pedido, '%Y-%m') as mes,
  sum(itens_pedido.quantidade * itens_pedido.preco_unitario) as valor_total
from
  pedidos
join
  itens_pedido on pedidos.pedido_id = itens_pedido.pedido_id
group by
  date_format(pedidos.data_pedido, '%Y-%m')
having
  valor_total > 1000;
```

A consulta tem como objetivo calcular o valor total das vendas por mês, filtrando apenas os meses em que o valor total das vendas ultrapassa 1000. Primeiramente, utiliza a função date_format() para extrair o ano e o mês da coluna data_pedido da tabela pedidos, agrupando os resultados por mês. Depois, é realizado um join entre as tabelas pedidos e itens_pedido, unindo-as pela coluna pedido_id. Em seguida, a soma do produto entre a quantidade e o preço unitário dos itens é calculada para obter o valor_total das vendas mensais. Por fim, o having é usada para filtrar os resultados e exibir apenas os meses em que o valor_total das vendas é superior a 1000.

```
select
  produtos.nome,
  categorias.nome
from
  produtos
inner join
  categorias on produtos.categoria_id = categorias.categoria_id
left join
  itens_pedido on produtos.produto_id = itens_pedido.produto_id and
itens_pedido.pedido_id IN (
    select pedido_id
    from pedidos
    where data_pedido >= curdate() - interval 90 day
  )
where
  itens_pedido.produto_id is null;
```

A consulta tem como objetivo listar os nomes de produtos e suas categorias que não foram vendidos nos últimos 90 dias. Primeiro, a consulta realiza um inner join entre as tabelas produtos e categorias para associar cada produto à sua categoria. Depois, é realizado um left join entre produtos e itens_pedido, com a condição de que apenas os pedidos realizados nos últimos 90 dias sejam pegos, utilizando uma query dentro do join. Por fim, o where filtra os resultados para incluir apenas os produtos que não possuem registros na tabela itens_pedido nos últimos 90 dias (indicados por itens_pedido.produto_id is null).

```
select
  cargo,
  count(*) as quantidade_funcionarios
from
  funcionarios
group by
  cargo
having
  count(*) > 2;
```

Essa consulta seleciona a coluna cargo e conta o número de funcionários para cada cargo, usando a função count(*) e renomeando essa contagem como quantidade_funcionarios. Os resultados são agrupados pela coluna cargo para obter a contagem de funcionários por cargo. Além disso, a consulta usa o having para filtrar os grupos e exibir apenas aqueles que tem mais de dois funcionários.

```
with pedidos_por_cliente as (
  select
    cliente_id,
    count(*) as total_pedidos
  from
    pedidos
  group by
    cliente_id
  having
    count(*) > 3),
pedidos_detalhados as (
  select
    p.cliente_id,
    p.funcionario_id,
    count(*) as pedidos_por_funcionario
  from
    pedidos p
```

```
  group by
    p.cliente_id, p.funcionario_id)
select
  c.nome as nome_cliente,
  f.nome as nome_funcionario,
  pd.pedidos_por_funcionario,
  ppc.total_pedidos
from
  pedidos_detalhados pd
inner join
  pedidos_por_cliente ppc on pd.cliente_id = ppc.cliente_id
inner join
  clientes c on pd.cliente_id = c.cliente_id
inner join
  funcionarios f on pd.funcionario_id = f.funcionario_id;
```

A consulta seleciona os nomes dos clientes e dos funcionários, além de detalhar o número de pedidos feitos por cada funcionário e o total de pedidos por cliente. A primeira parte pedidos_por_cliente conta o número de pedidos por cliente, filtrando apenas os clientes que fizeram mais de 3 pedidos. A segunda parte pedidos_detalhados conta o número de pedidos feitos por cada funcionário para cada cliente. A parte final junta essas as duas partes iniciais com as tabelas clientes e funcionarios para obter o que foi pedido.

```
select
  clientes.nome as nome_cliente,
  produtos.nome as nome_produto,
  itens_pedido.quantidade,
  itens_pedido.preco_unitario,
  pedidos.data_pedido
from
  clientes
inner join
  pedidos on clientes.cliente_id = pedidos.cliente_id
inner join
  itens_pedido on pedidos.pedido_id = itens_pedido.pedido_id
inner join
  produtos on itens_pedido.produto_id = produtos.produto_id
where
  clientes.cliente_id = 5
  and pedidos.data_pedido >= curdate() - interval 1 year
order by
  pedidos.data_pedido;
```

Essa consulta seleciona os nomes dos clientes e produtos, além da quantidade e preço unitário dos itens pedidos e a data do pedido. Primeiro, a consulta faz um inner join entre as tabelas clientes, pedidos, itens_pedido e produtos, associando os clientes aos seus pedidos, os pedidos aos itens pedidos e os itens pedidos aos produtos. O where filtra para incluir apenas os pedidos feitos pelo cliente com cliente_id = 5 no último ano, utilizando a função curdate() e o interval. No final, os resultados são ordenados pela data do pedido.

```
select
  produtos.nome,
  sum(itens_pedido.preco_unitario * itens_pedido.quantidade) as receita_total
from
  itens_pedido
inner join
  produtos on itens_pedido.produto_id = produtos.produto_id
group by
  produtos.nome
order by
  receita_total desc
limit 1;
```

A consulta seleciona os nomes dos produtos e calcula a receita total gerada por cada produto. Na primeira parte, é realizado um inner join entre as tabelas itens_pedido e produtos para associar cada item pedido ao seu produto. Em seguida, a função sum() calcula a receita total de cada produto, multiplicando o preço unitário pela quantidade dos itens vendidos. A consulta agrupa os resultados pelo nome dos produtos e ordena esses resultados em ordem decrescente de receita total. Por fim, o limit 1 é usada para retornar apenas o produto com a maior receita total.

Parte 4: Alteração e Manutenção no Banco de Dados

```
alter table
  produtos
add
  desconto decimal(10,2);
```

A consulta altera a estrutura da tabela produtos. O alter table é utilizado para modificar a definição da tabela existente. O add é utilizado para adicionar uma nova coluna chamada desconto à tabela produtos. A nova coluna desconto é do tipo decimal com precisão de 10 dígitos, sendo 2 deles após a vírgula decimal.

```
alter table
  funcionarios
add
  cpf int;
```

A consulta altera a estrutura da tabela funcionarios. O alter table é utilizado para modificar a tabela. O add é utilizado para adicionar uma nova coluna chamada cpf à tabela funcionarios. A nova coluna cpf é do tipo inteiro (int).

Parte 5: Diagrama de Relacionamento de Entidades