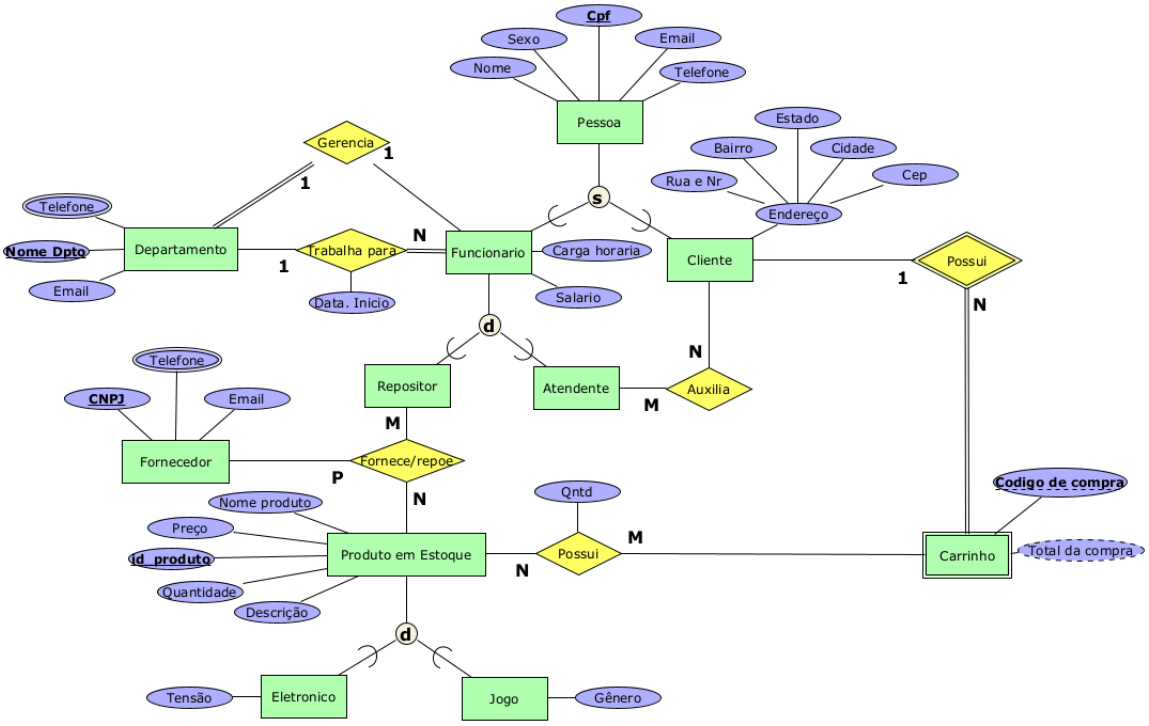
Lavras — MG
2022

Descrição:

Recolhimento de informações para o sistema de um E-commerce de eletrônicos chamado Drexel. As relações do E-commerce são repartidas em 'Funcionários' e 'Clientes', ambos membros da superclasse 'Pessoa'. Funcionários trabalham para diferentes departamentos e são divididos em 'atendentes' e 'repositores'. Em relação aos atributos, o 'id' de um produto e o 'código de compra' de um determinado cliente são de extrema importância. Tanto para funcionários como para clientes a chave tida como primária é o CPF. O CPF é utilizado como a chave primária que completa a chave parcial 'Código de compra' contida na entidade fraca 'Carrinho', para identificar o proprietário dos itens. Cada entidade possui seus próprios atributos individuais, mas 'Email', 'Sexo', 'CPF', 'Nome', 'Telefone' são alguns atributos comuns a 'Clientes' e 'Funcionários', por exemplo. A entidade 'Carrinho' se relaciona com a entidade 'Produto em Estoque', pois é necessário verificar se um determinado item se encontra no estoque, para poder pertencer ao carrinho. Há um relacionamento ternário entre 'Repositor', 'Fornecedor' e 'Produto em Estoque', Fornecedor disponibiliza o produto para ser reposto no estoque. Todo funcionário trabalha obrigatoriamente para um departamento (1:N Duplo), sendo somente um funcionário responsável por gerenciar o departamento como um todo.
(Obs : Optamos por não adicionar um atributo 'preço comprado' no relacionamento 'Possui' pois 'Total da Compra' (atributo derivado) já é responsável por armazenar esta informação).

.

Diagrama ER



Dicionário Tipos Entidades:

01)

Tipo Entidade	**Pessoa**		
Descrição	Atributos em comum das entidades cliente e funcionário, pessoa pode ser ou um funcionário, ou um cliente, ou mesmo ambos ao mesmo tempo, qualificando uma restrição de sobreposição.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
CPF	Cadastro de pessoa física.	Texto(11) Formato: ddd.ddd.ddd-dd	N
Nome	Nome da pessoa.	Texto(50)	N
Telefone	Telefone da pessoa.	Texto(11) Formato: (dd)ddddd-dddd	N
Sexo	Sexo (masculino/feminino).	Texto(1) M - Masculino F - Feminino	N
Email	Endereço de e-mail.	Texto(35)	N

02)

Tipo Entidade	Cliente		
Descrição	Subclasse sobreposta da superclasse do tipo entidade “Pessoa”, onde são alocados os clientes.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Endereço	Endereço físico da pessoa.	Texto(80)	N
CEP	Código de endereçamento postal.	Texto(8) Formato: ddddd-ddd	N

03)

Tipo Entidade	Funcionário
Descrição	Subclasse sobreposta da superclasse do tipo entidade “Pessoa”, onde são alocados os funcionários que ali trabalham. A subclasse, no entanto, possui outras duas

	subclasses (Repositor e Atendente) as quais possuem restrição de disjunção. Suas subclasses representam duas especializações de funcionários dentro da empresa.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Salário	Salário do funcionário	Real(6,2) positivo	N
Carga horária	Horas trabalhadas pelo funcionário.	Real(3,2) positivo	N

04)

Tipo Entidade	Departamento		
Descrição	Representação dos departamentos da empresa Drexel, cujas estruturas tem o intuito de organizar os funcionários contratados da empresa.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Nome	Nome do departamento.	Texto(40)	N
Telefone	Telefone do departamento.	Texto(11) Formato: (dd)ddddd-dddd	N
Email	Endereço de e-mail.	Texto(35)	N

05)

Tipo Entidade	Repositor		
Descrição	Subclasse que representa um tipo de especialização de funcionário com restrição de disjunção.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**

06)

Tipo Entidade	Atendente		
Descrição	Subclasse que representa um tipo de especialização de funcionário com restrição de disjunção.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo?**

			(S/N)

07)

Tipo Entidade	Produtos em Estoque		
Descrição	Conjunto de produtos que serão repostos pelo repositor da empresa Drexel. Representa uma “superclasse” dentro da hierarquia que participa e possui como subclasses disjuntas os tipos entidade: Eletrônico e Jogos (as quais não possuem relacionamentos).		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
ID do produto	Identificador do produto para facilitar as buscas.	Inteiro(6) positivo	N
Preço	Valor por cada unidade do produto.	Real(8,2) positivo	N
Nome Produto	Nome de identificação do produto.	Texto(40)	N
Quantidade	Quantidade de produtos em estoque.	Inteiro(6) positivo	N
Descrição	Descrição detalhada do produto	Texto(100)	N

08)

Tipo Entidade	Fornecedor		
Descrição	Lista de fornecedores de produtos para a empresa Drexel.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
CNPJ	Cadastro de pessoa jurídica da empresa.	Texto(14) Formato: dd.ddd.ddd/dddd-00	N
Telefone	Telefone do fornecedor em questão.	Texto(11) Formato: (dd)ddddd-dddd	N
Email	Endereço de e-mail.	Texto(35)	N

09)

Tipo Entidade	Carrinho

Descrição	É o local onde os clientes e visitantes da empresa Drexel organizam os produtos os quais pretendem comprar na loja.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Código de compra	Código identificador de cada compra realizada pelo usuário.	Inteiro(5) positivo	N
Total da compra	Valor somado de todas unidades do produto selecionado.	Real(8,2) positivo	N

10)

Tipo Entidade	Eletrônico		
Descrição	Conjunto de produtos eletrônicos disponibilizados para compra pela empresa Drexel.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Tensão/DDP	Informa qual a tensão necessária para ligar o aparelho na tomada	Inteiro(3) positivo	N

11)

Tipo Entidade	Jogos		
Descrição	Conjunto de jogos disponíveis em estoque que podem ser comprados pelos clientes e visitantes da empresa.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Gênero	Categorização dos jogos que pode variar de acordo com os aspectos diferenciados entre eles. Exemplos: terror, RPG, Ação, entre outros.	Texto(30)	N

Dicionário Tipo Relacionamentos:

01)

Tipo Relacionamento	Trabalha-para
Descrição	Informa em qual departamento o funcionário é alocado para trabalhar.
Atributos	

Nome	Descrição	Domínio	Permite nulo? (S/N)
Data início	Indica a data em que um funcionário começou a trabalhar para um determinado departamento	Data	N

02)

Tipo Relacionamento	Gerencia
Descrição	Indica a relação de um funcionário que ocupa o cargo de gerente e qual departamento ele gerencia.

03)

Tipo Relacionamento	Possui		
Descrição	Informa a quantidade e os itens que estão no carrinho.		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Quantidade	Número de unidades do item selecionados para adicionar no carrinho	Inteiro(4) positivo	N

04)

Tipo Relacionamento	Auxilia
Descrição	Indica o atendimento de um determinado funcionário aos clientes.

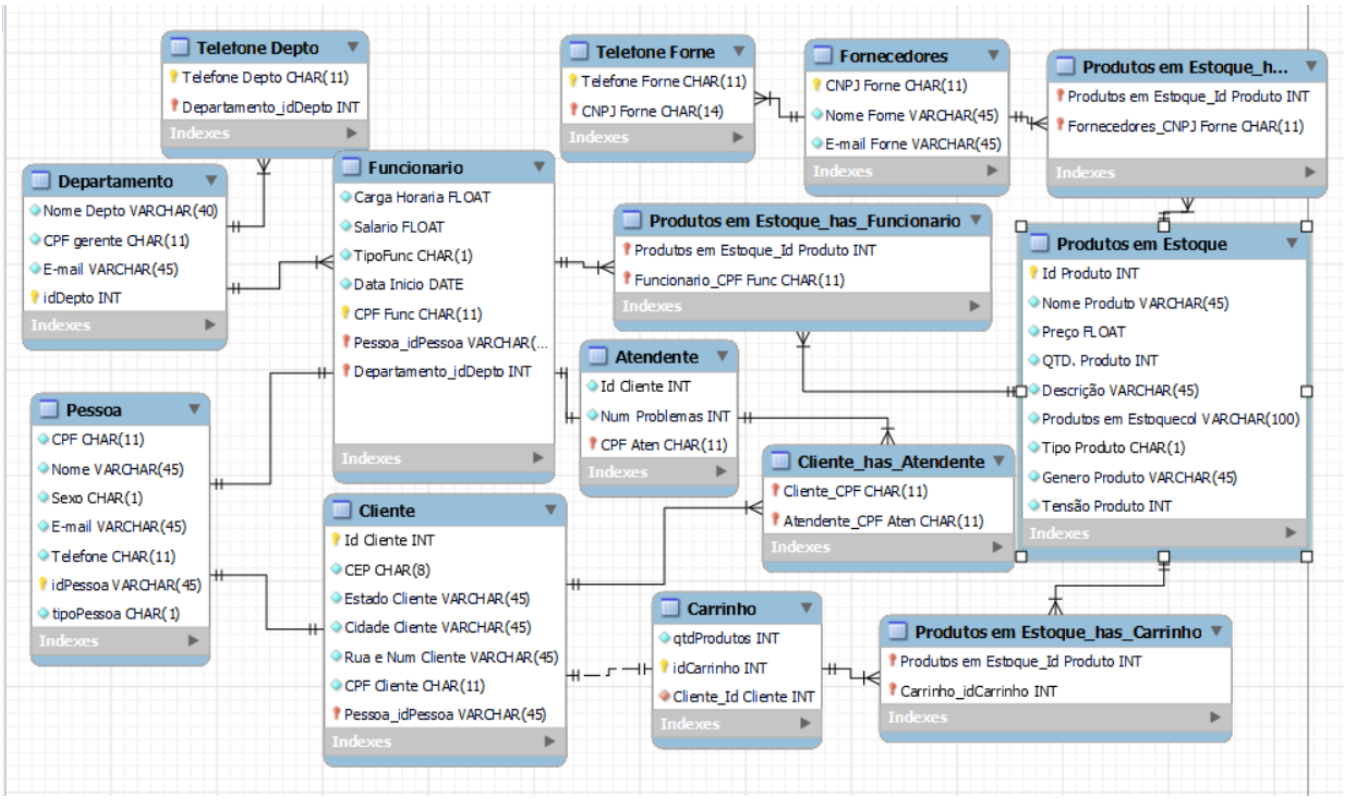
05)

Tipo Relacionamento	Fornece/Repõe
Descrição	Representa o fornecimento e a reposição de determinados produtos.

06)

Tipo Relacionamento	Possui
Descrição	Relacionamento identificador que indica quais itens do estoque o carrinho de um determinado cliente possui.

Diagrama Relacional (Corrigido e com as opções de exclusão)



Dicionário de dados – Relacional

Tabela	Pessoa
Descrição	Atributos em comum das entidades cliente e funcionário, pessoa pode ser ou um funcionário, ou um cliente, ou mesmo ambos ao mesmo tempo, qualificando uma restrição de sobreposição.
Atributos	
Nome	**Descrição.**
CPF	Cadastro de pessoa física. Chave primária.
Nome	Nome da pessoa.
Telefone	Telefone da pessoa.
Sexo	Masculino ou Feminino.
Email	Endereço de e-mail da pessoa.
Id Pessoa	Valor único atribuído a pessoa.
Tipo Pessoa	Determina se a pessoa será um cliente ou funcionário

Tabela	Cliente
Descrição	Subclasse sobreposta da superclasse do tipo entidade “Pessoa”, onde são alocados os clientes.
Atributos	
Nome	**Descrição.**
IdCliente	Código individual do cliente
Estado Cliente	Estado de moradia do cliente
Cidade Cliente	Cidade de moradia do cliente
Rua e Num. Cliente	Rua e Numero de residencia do cliente
CEP	Código de endereçamento postal.
CPF	Referência a tabela Pessoa.
Pessoa idPessoa	Referência idPessoa da tabela Pessoa

Tabela	Funcionário
Descrição	Subclasse sobreposta da superclasse do tipo entidade “Pessoa”, onde são alocados os funcionários que ali trabalham. A subclasse, no entanto, possui outras duas subclasses (Repositor e Atendente) as quais possuem restrição de disjunção. Suas subclasses representam duas especializações de funcionários dentro da empresa.
Atributos	
Nome	**Descrição.**
CargaHoraria	Horas de jornada de trabalho do funcionário.
Salário	Salário referente ao funcionário.
Tipo Func	Determina se o tipo do funcionário, A = Atendente, V = Vendedor.
Data Inicio	Data que iniciou sua contratação
CPF Func	CPF do funcionário em questão.
Pessoa_idPessoa	Referência idPessoa na tabela Pessoa
Departamento_idDepartamento	Referência idDepartamento na tabela Departamento

Tabela	Departamento
Descrição	Subclasse que representa um tipo de especialização de funcionário com restrição de disjunção.
Atributos	
Nome	**Descrição.**
Nome	Descrição.
CPF gerente	Número do cpf do gerente do departamento
Email	Endereço de e-mail do departamento.
idDepartamento	Número de id do departamento.

Tabela	Telefone Depto
Descrição	Local o qual se armazena os telefones dos departamentos.
Atributos	
Nome	**Descrição.**

Tabela	Telefone Depto
Descrição	Local o qual se armazena os telefones dos departamentos.
Atributos	
Telefone Depto	Representa o número de telefone do departamento.
Departamento_idDepartamento	Referência idDepartamento da tabela Departamento

Tabela	Atendente
Descrição	Subclasse que representa um tipo de especialização de funcionário com restrição de disjunção.
Atributos	
Nome	**Descrição.**
NumProblemas	Número total de atendimentos convertidos e solucionados por esse atendente durante um mês.
CPF Aten	Referência a tabela Funcionario.
Id Cliente	Identificação do cliente atendido.

Tabela	Cliente_has_Atendente
Descrição	Representa o vínculo entre cliente e atendente, visando armazenar qual atendente prestou serviços a qual cliente.
Atributos	
Nome	**Descrição.**
Cliente_CPF	Número de CPF atrelado ao cliente, referência CPF Cliente na tabela Cliente.
Atendente_CPF Aten	Referência o CPF Aten da tabela atendente, visando armazenar o cpf do atendente que prestou o serviço.

Tabela	ProdutosEstoque
Descrição	Conjunto de produtos que serão repostos pelo repositor da empresa Drexel. Representa uma “superclasse” dentro da hierarquia que participa e possui como subclasses disjuntas os tipos entidade: Eletrônico e Jogos (as quais não possuem relacionamentos).

Atributos	
Nome	**Descrição.**
IdProduto	Identificador individual do produto. Chave Primária.
Nome Produto	Nome do produto.
Preço	Valor por unidade do produto.
QTD. Produto	Quantidade do produto em estoque.
Descrição	Descrição detalhada do produto.
TipoProduto	Determina o tipo de produto. E = Eletrônicos, J = Jogos e O = Outros.
Produtos em Estoque	Quantidade de produtos no estoque.
Gênero Produto	Dependente do TipoProduto caso seja J, essa coluna é preenchida.
Tensão	Dependente da TipoProduto caso seja E, essa coluna é preenchida.

Tabela	Fornecedores
Descrição	Lista de fornecedores de produtos para a empresa Drexel.
Atributos	
Nome	**Descrição.**
Nome	Descrição.
CNPJ Forne	Cadastro de pessoa jurídica da empresa.
Telefone Forne	Telefone referente ao fornecedor.
Nome	Nome do fornecedor.

Tabela	Telefone Forne
Descrição	Armazena os números de telefone atrelados a um fornecedor.
Atributos	
Nome	**Descrição.**
Telefone Forne	Número de telefone do fornecedor em questão.
CNPJ Do Forne	Número do CNPJ do fornecedor.

Tabela	Produtos em Estoque_has_Fornecedores
Descrição	Representa o vínculo entre as duas tabelas em questão, Produtos em Estoque e Fornecedores.
Atributos	
Nome	**Descrição.**
Produtos em Estoque_Id Produto	Referência o idProduto da tabela Produtos em Estoque.
Fornecedores_CNPJ Forne	Referencia o CNPJ da tabela Fornecedores.

Tabela	Produtos em Estoque_has_Funcionario
Descrição	Representa o vínculo entre as duas tabelas em questão, Produtos em Estoque e Funcionário.
Atributos	
Nome	**Descrição.**
Produtos em Estoque_Id Produto	Referência o idProduto da tabela Produtos em Estoque.
Funcionario_CPF Func	Referencia o CPF Func da tabela Funcionario.

Tabela	Carrinho
Descrição	É o local onde os clientes e visitantes da empresa Drexel organizam os produtos os quais pretendem comprar na loja.
Atributos	
Nome	**Descrição.**
qtdProdutos	Numero de produtos dentro do carrinho.
idCarrinho	Valor único atribuído ao carrinho.
Cliente_idCliente	Referência idCliente na tabela

Tabela	Carrinho
Descrição	É o local onde os clientes e visitantes da empresa Drexel organizam os produtos os quais pretendem comprar na loja.
Atributos	
Nome	**Descrição.**
qtdProdutos	Numero de produtos dentro do carrinho.
idCarrinho	Valor único atribuído ao carrinho.
Cliente_idCliente	Referência idCliente na tabela

Tabela	Produtos em Estoque_has_Carrinho
Descrição	Representa o vínculo entre as duas tabelas em questão, Produtos em Estoque e Carrinho.
Atributos	
Nome	**Descrição.**
Produtos em Estoque_Id Produto	Referência o idProduto da tabela Produtos em Estoque.
Carrinho_idCarrinho	Referencia o idCarrinho da tabela Carrinho.

Etapa 3 - Scripts SQL

b)

```sql
DROP TABLE if exists `mydb`.`teste`;

CREATE TABLE `mydb`.`teste` (
  `idteste` INT NOT NULL AUTO_INCREMENT,
  `Nome_tester` VARCHAR(50) NOT NULL,
  `Endereco` VARCHAR(50) NOT NULL,
  `Cidade` VARCHAR(45) NOT NULL,
  `Estado` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idteste`));

Alter table `mydb`.`teste`
add column `limite_de_credito` decimal(10,2) not null;

ALTER TABLE `mydb`.`teste`
MODIFY COLUMN `Limite_de_credito` INT NOT NULL;

ALTER TABLE `mydb`.`teste`
DROP COLUMN `Limite_de_credito`;
```

c)

```sql
-- Inserção de valores na tabela "Pessoa":
INSERT INTO `mydb`.pessoa(CPF, Nome, Sexo, `E-mail`, Telefone, idPessoa, tipoPessoa)
VALUES (12345678911, 'F1', 'M', 'teste@gmail.com', '35999999999', 1, 'f'),
       (12345678912, 'F2', 'M', 'teste@gmail.com', '35999999999', 2, 'f'),
       (12345678913, 'F3', 'F', 'teste@gmail.com', '35999999999', 3, 'f'),
       (12345678914, 'F4', 'M', 'teste@gmail.com', '35999999999', 4, 'f'),
       (12345678915, 'F5', 'F', 'teste@gmail.com', '35999999999', 5, 'f'),
       (12345678916, 'F6', 'F', 'teste@gmail.com', '35999999999', 6, 'f'),
       (12345678917, 'F7', 'M', 'teste@gmail.com', '35999999999', 7, 'f'),
       (12345678918, 'F8', 'M', 'teste@gmail.com', '35999999999', 8, 'f'),
       (12345678919, 'F9', 'F', 'teste@gmail.com', '35999999999', 9, 'f'),
       (12345678921, 'F10', 'M', 'teste@gmail.com', '35999999999', 10, 'f'),
       (12345678922, 'F11', 'F', 'teste@gmail.com', '35999999999', 11, 'f'),
       (12345678923, 'F12', 'F', 'teste@gmail.com', '35999999999', 12, 'f'),
       (12345678924, 'F13', 'M', 'teste@gmail.com', '35999999999', 13, 'f'),
       (12345678925, 'F14', 'M', 'teste@gmail.com', '35999999999', 14, 'f'),
       (12345678926, 'F15', 'F', 'teste@gmail.com', '35999999999', 15, 'f'),
       (12345678927, 'F16', 'M', 'teste@gmail.com', '35999999999', 16, 'f'),
       (12345678928, 'F17', 'F', 'teste@gmail.com', '35999999999', 18, 'f'),
       (12345678931, 'F18', 'M', 'teste@gmail.com', '35999999999', 19, 'f'),
       (12345678932, 'F19', 'M', 'teste@gmail.com', '35999999999', 20, 'f'),
       (12345678933, 'C1', 'M', 'teste@gmail.com', '35999999999', 21, 'c'),
       (12345678934, 'C2', 'M', 'teste@gmail.com', '35999999999', 22, 'c'),
       (12345678935, 'C3', 'F', 'teste@gmail.com', '35999999999', 23, 'c'),
       (12345678936, 'C4', 'M', 'teste@gmail.com', '35999999999', 24, 'c'),
       (12345678937, 'C5', 'F', 'teste@gmail.com', '35999999999', 25, 'c'),
       (12345678938, 'C6', 'F', 'teste@gmail.com', '35999999999', 26, 'c'),
       (12345678939, 'C7', 'M', 'teste@gmail.com', '35999999999', 27, 'c'),
       (12345678940, 'C8', 'M', 'teste@gmail.com', '35999999999', 28, 'c'),
       (12345678941, 'C9', 'F', 'teste@gmail.com', '35999999999', 29, 'c'),
       (12345678942, 'C10', 'M', 'teste@gmail.com', '35999999999', 30, 'c'),
       (12345678943, 'C11', 'F', 'teste@gmail.com', '35999999999', 31, 'c'),
       (12345678944, 'C12', 'F', 'teste@gmail.com', '35999999999', 32, 'c'),
       (12345678945, 'C13', 'M', 'teste@gmail.com', '35999999999', 33, 'c'),
       (12345678946, 'C14', 'M', 'teste@gmail.com', '35999999999', 34, 'c'),
       (12345678947, 'C15', 'F', 'teste@gmail.com', '35999999999', 35, 'c'),
       (12345678948, 'C16', 'M', 'teste@gmail.com', '35999999999', 36, 'c'),
       (12345678949, 'C17', 'F', 'teste@gmail.com', '35999999999', 37, 'c'),
       (12345678950, 'C18', 'M', 'teste@gmail.com', '35999999999', 38, 'c');

-- Inserção de valores na tabela Departamento
INSERT INTO `mydb`.departamento(`Nome Depto`, `CPF gerente`, `E-mail`, idDepto)
VALUES ('Vendas', 12345678911, 'deptovendas@gmail.com', 1),
       ('RH', 12345678912, 'deptorh@gmail.com', 2),
       ('Recepcao', 12345678913, 'deptorec@gmail.com', 3),
       ('Administracao', 12345678914, 'deptoadm@gmail.com', 4),
       ('Estoque', 12345678915, 'deptoest@gmail.com', 5),
       ('ex1', 12345678916, 'deptoex1@gmail.com', 6),
       ('ex2', 12345678917, 'deptoex2@gmail.com', 7),
       ('ex3', 12345678918, 'deptoex3@gmail.com', 8),
       ('ex4', 12345678919, 'deptoex4@gmail.com', 9),
       ('ex5', 12345678921, 'deptovex5@gmail.com', 10);
```

```sql
-- Inserção de valores na tabela Telefone Departamento
-- OBS: A inserção só é possível se o departamento já existir
INSERT INTO `mydb`.`telefone depto`(`Telefone Depto`, Departamento_idDepto)
VALUES ('35999999999', 1), ('35999999999', 2),
       ('35999999999', 3), ('35999999999', 4),
       ('35999999999', 5), ('35999999999', 6),
       ('35999999999', 7), ('35999999999', 8),
       ('35999999999', 9), ('35999999999', 10);

-- Inserção de valores na tabela Funcionário
INSERT INTO `mydb`.funcionario(`Carga Horaria`, Salario, TipoFunc, `Data Inicio`,
                               `CPF Func`, Pessoa_idPessoa, `Departamento_idDepto`)
VALUES (1, 10000, 'V', '2020-01-01', 12345678911, 1, 1),
       (1, 10000, 'A', '2020-01-01', 12345678913, 3, 1),
       (1, 10000, 'V', '2023-01-01', 12345678915, 5, 3),
       (1, 10000, 'A', '2023-01-01', 12345678917, 7, 4),
       (1, 10000, 'V', '2023-01-01', 12345678919, 9, 5),
       (1, 10000, 'A', '2023-01-01', 12345678922, 11, 6),
       (1, 10000, 'V', '2023-01-01', 12345678924, 13, 7),
       (1, 10000, 'A', '2023-01-01', 12345678926, 15, 8),
       (1, 10000, 'V', '2023-01-01', 12345678928, 18, 9),
       (1, 10000, 'A', '2023-01-01', 12345678932, 20, 10),
       (1, 1000, 'A', '2020-01-01', 12345678912, 2, 1),
       (1, 1000, 'V', '2023-01-01', 12345678914, 4, 2),
       (1, 1000, 'A', '2023-01-01', 12345678916, 6, 3),
       (1, 1000, 'V', '2023-01-01', 12345678918, 8, 4),
       (1, 1000, 'A', '2023-01-01', 12345678921, 10, 5),
       (1, 1000, 'V', '2023-01-01', 12345678923, 12, 6),
       (1, 1000, 'A', '2023-01-01', 12345678925, 14, 7),
       (1, 1000, 'V', '2023-01-01', 12345678927, 16, 8),
       (1, 1000, 'A', '2023-01-01', 12345678931, 19, 9);
```

```sql
-- Inserção de valores na tabela Atendente
INSERT INTO `mydb`.atendente(`Id Cliente`, `Num Problemas`, `CPF Aten`)
VALUES (1, 10, 12345678913), (2, 5, 12345678917), (3, 18, 12345678922), (4, 12, 12345678926), (5, 32, 12345678932);
```

```sql
-- Inserção de valores na tabela Cliente
INSERT INTO `mydb`.cliente(`Id Cliente`, CEP, `Estado Cliente`, `Cidade Cliente`,
                           `Rua e Num Cliente`, `CPF Cliente`, Pessoa_idPessoa)
VALUES (1,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678933, 21),
       (2,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678934, 22),
       (3,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678935, 23),
       (4,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678936, 24),
       (5,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678937, 25),
       (6,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678938, 26),
       (7,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678939, 27),
       (8,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678940, 28),
       (9,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678941, 29),
       (10, 11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678942, 30),
       (11,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678943, 31),
       (12,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678944, 32),
       (13, 11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678945, 33),
       (14,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678946, 34),
       (15,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678947, 35),
       (16,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678948, 36),
       (17,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678949, 37),
       (18,  11111111, 'Minas Gerais', 'Lavras', 'AAAAAA', 12345678950, 38);


-- Inserção de valores na tabela Cliente_Has_Atendente
INSERT INTO `mydb`.cliente_has_atendente(Cliente_CPF, `Atendente_CPF Aten`)
VALUES (12345678933, 12345678913), (12345678934, 12345678917), (12345678935, 12345678922),
       (12345678936, 12345678926), (12345678937, 12345678932);

/* Ou seja, cliente C1 foi atendido pelo funcionario F3) - Primeiro valor inserido, exemplo */


-- Inserção de valores em Carrinho
INSERT INTO mydb.Carrinho(qtdProdutos, idCarrinho, `Cliente_Id Cliente`)
VALUES (10,  1,  1), (10,  2,  2), (10,  3,  3), (10,  4,  4), (10,  5,  5), (10,  6,  6),
       (10,  7,  7), (10,  8,  8), (10,  9,  9), (10, 10, 10), (10, 11, 11), (10, 12, 12),
       (10, 13, 13), (10, 14, 14), (10, 15, 15), (10, 16, 16), (10, 17, 17), (10, 18, 18);
```

```sql
-- Inserção de valores na tabela Fornecedores
INSERT INTO `mydb`.fornecedores(`CNPJ Forne`, `Nome Forne`, `E-mail Forne`)
VALUES (12345678910, 'Fornecedor1', 'fornecedor1@gmail.com'),
       (12345678911, 'Fornecedor2', 'fornecedor2@gmail.com'),
       (12345678912, 'Fornecedor3', 'fornecedor3@gmail.com'),
       (12345678913, 'Fornecedor4', 'fornecedor4@gmail.com'),
       (12345678914, 'Fornecedor5', 'fornecedor5@gmail.com'),
       (12345678915, 'Fornecedor6', 'fornecedor6@gmail.com');
```

```sql
-- Inserção de valores na tabela Telefone Fornecedores
INSERT INTO `mydb`.`telefone forne`(`Telefone Forne`, `CNPJ Forne`)
VALUES (35999999999, 12345678910), (35999999999, 12345678911),
       (35999999999, 12345678912), (35999999999, 12345678913),
       (35999999999, 12345678914), (35999999999, 12345678915);
```

```sql
-- Inserção de valores na tabela Produtos em Estoque
INSERT INTO `mydb`.`produtos em estoque`(`Id Produto`, `Nome Produto`,
                                         Preço, `QTD. Produto`, Descrição,
                                         `Produtos em Estoquecol`,`Tipo Produto`,
                                         `Genero Produto`, `Tensão Produto`,
                                         `Produtos em Estoquecol1`)

VALUES (1, 'ProdutoA', 1000, 100, 'Produto teste 01', 50, 'J', 'Ação', 220, 25),
       (2, 'ProdutoB', 10000, 100, 'Produto teste 02', 50, 'J', 'Terror', 220, 25),
       (3, 'ProdutoC', 2000, 100, 'Produto teste 03', 50, 'J', 'Ação', 220, 25),
       (4, 'ProdutoD', 20000, 100, 'Produto teste 04', 50, 'J', 'Terror', 220, 25),
       (5, 'ProdutoE', 3000, 100, 'Produto teste 05', 50, 'J', 'Ação', 220, 25),
       (6, 'ProdutoF', 30000, 100, 'Produto teste 06', 50, 'J', 'Terror', 220, 25),
       (7, 'ProdutoG', 4000, 100, 'Produto teste 07', 50, 'J', 'Ação', 220, 25),
       (8, 'ProdutoH', 40000, 100, 'Produto teste 08', 50, 'J', 'Terror', 220, 25),
       (9, 'ProdutoI', 1500, 100, 'Produto teste 09', 50, 'J', 'Ação', 220, 25),
       (10, 'ProdutoJ', 2500, 100, 'Produto teste 10', 50, 'J', 'Terror', 220, 25),
       (11, 'ProdutoK', 3500, 100, 'Produto teste 11', 50, 'J', 'Ação', 220, 25),
       (12, 'ProdutoL', 4500, 100, 'Produto teste 12', 50, 'J', 'Terror', 220, 25);
```

```sql
-- Inserção de valores na tabela Produtos Em Estoque_Has_Carrinho
INSERT INTO `mydb`.`produtos em estoque_has_carrinho`(`Produtos em Estoque_Id Produto`,
                                                      Carrinho_idCarrinho)

VALUES (1, 1), (1, 2), (1, 3), (2, 1), (3, 1);

/* Permite a inserção de um mesmo produto em vários
carrinhos e de vários produtos num mesmo carrinho*/


-- Inserção de valores na tabela Produtos Em Estoque_Has_Fornecedores
INSERT INTO `mydb`.`produtos em estoque_has_fornecedores`(`Produtos em Estoque_Id Produto`,
                                                           `Fornecedores_CNPJ Forne`)

VALUES (1, 12345678910), (1, 12345678911),(2, 12345678910);

/* Um mesmo produto pode ser fornecido por diferentes fornecedores
   e um mesmo fornecedor pode fornecer vários produtos */

-- Inserção de valores na tabela Produtos em estoque_Has_Funcionario
INSERT INTO `mydb`.`produtos em estoque_has_funcionario`(`Produtos em Estoque_Id Produto`,
                                                          `Funcionario_CPF Func`)

VALUES (1, 12345678911), (2, 12345678911), (1, 12345678913);
/* Um mesmo produto pode estar relaciona a mais de um funcionario e
um mesmo funcionario pode estar relacionado a mais de um produto */
```

d)

```sql
UPDATE Cliente SET `Estado Cliente` = 'SP' WHERE `Id Cliente` = 1;

UPDATE Cliente SET `Cidade Cliente` = 'los angeles' WHERE `CPF Cliente` = '11111111111';

/* Aumenta em 10% o salario de todos os funcionarios que possuam mais de dois produtos em estoque */
UPDATE `mydb`.funcionario f
SET f.Salario = f.Salario + (f.Salario*0.1)
WHERE f.`CPF Func` IN (SELECT mef.`Funcionario_CPF Func`
    FROM `mydb`.`produtos em estoque_has_funcionario` mef
    WHERE mef.`Produtos em Estoque_Id Produto` > 1);
```

```sql
/*Atualiza o email do getente cujo departamento é o RH (Departamento):*/
UPDATE Departamento
SET `E-mail` = 'new_email@example.com'
WHERE `CPF gerente` = (
  SELECT CPF FROM Pessoa
  WHERE `Nome` = 'John Doe'
  AND `tipoPessoa` = 'G')

  AND `Nome Depto` = 'RH';

/**/
UPDATE `mydb`.`produtos em estoque`
SET Preço = Preço + (Preço*0.25)
WHERE `Tipo Produto` = 'J' AND `Genero Produto` = 'Ação';
```

e)

```sql
/*Observe que o código utiliza várias subconsultas para localizar os registros
que devem ser excluídos. Além disso, é necessário realizar a exclusão em uma
ordem específica, para evitar violações de integridade referencial
(por exemplo, não é possível excluir um departamento se ele ainda tiver
funcionários associados a ele).*/

DELETE FROM `Telefone Depto` WHERE Departamento_idDepto IN (
  SELECT Departamento_idDepto FROM Funcionario WHERE Pessoa_idPessoa IN (
    SELECT Pessoa_idPessoa FROM Cliente WHERE `Id Cliente` = 1
  )
);

/*exclui os funcionarios do cliente*/
SET FOREIGN_KEY_CHECKS = 0;
DELETE FROM `mydb`.funcionario
WHERE Pessoa_idPessoa IN (SELECT idPessoa
                          FROM `mydb`.pessoa
                          WHERE Nome = 'F3');
```

```sql
/*Exclui as pessoas que são funcionários dos departamentos gerenciados pelo mesmo
gerente do cliente que está sendo excluído*/

DELETE FROM Pessoa WHERE idPessoa IN (
  SELECT Pessoa_idPessoa FROM Funcionario WHERE Departamento_idDepto IN (
    SELECT idDepto FROM Departamento WHERE `CPF gerente` = (
      SELECT CPF Cliente FROM Cliente WHERE `Id Cliente` = 1
    )
  )
);

/*Exclui os departamentos gerenciados pelo mesmo gerente do cliente que está sendo excluído.*/

DELETE FROM Departamento WHERE `CPF gerente` = (
  SELECT `CPF Cliente` FROM Cliente WHERE `Id Cliente` = 1
);
-- autoexplicativo
DELETE FROM Cliente WHERE `Id Cliente` = 1;
```

f)

```sql
/* Consulta 01: Seleciona o nome e salario de todos os funcionarios */
-- Consulta mais básica, apenas para SELECT FROM WHERE
SELECT Salario, Nome
FROM funcionario JOIN pessoa
WHERE `CPF Func`= CPF;

/* Consulta 02: Selecione o nome e o salario de todos os funcionarios
cuja data de início esteja entre 2020 e 2022 */

SELECT Salario, Nome
FROM funcionario JOIN pessoa
WHERE funcionario.`CPF Func`= CPF AND `Data Inicio` BETWEEN "2020/01/01" AND "2022/12/31";


/* Consulta 03: Selecione, em ordem alfabetica, o nome dos funcionarios que trabalham
ou para o departamento de vendas ou para o departamento RH*/

SELECT funcionario.`CPF Func`, pessoa.Nome
FROM funcionario JOIN departamento JOIN pessoa
WHERE funcionario.Departamento_idDepto = departamento.idDepto AND
(departamento.`Nome Depto` = "Vendas" OR departamento.`Nome Depto` = "RH") AND
pessoa.CPF = funcionario.`CPF Func`
ORDER BY pessoa.Nome;
```

```sql
/* Consulta 04: Recupere, em ordem alfabetica e para cada departamento, o seu nome,
número de funcionários, e seu salario médio*/

SELECT departamento.`Nome Depto`, COUNT(*) AS qtdFuncionario,
                                            AVG(funcionario.Salario)
                                            AS salarioMedio
FROM departamento JOIN funcionario
WHERE funcionario.Departamento_idDepto = departamento.idDepto
GROUP BY funcionario.Departamento_idDepto
ORDER BY departamento.`Nome Depto`;
```

```sql
/* Consulta 05: Recupere o nome e a quantidade de produtos cujo os funcionarios
tem em estoque, mas apenas para aqueles que tenham mais de um produto em seu estoque */

SELECT pessoa.Nome,
    COUNT(`produtos em estoque_has_funcionario`.`Funcionario_CPF Func`) AS qtdProdutos

FROM `produtos em estoque_has_funcionario` join funcionario join pessoa
WHERE `produtos em estoque_has_funcionario`.`Funcionario_CPF Func` = funcionario.`CPF Func`
    AND funcionario.`CPF Func` = pessoa.CPF

GROUP BY `produtos em estoque_has_funcionario`.`Funcionario_CPF Func`
HAVING qtdProdutos > 1;
```

```sql
/* Consulta 06: Recupere os dados de todas as pessoas que são funcionárias. No caso de existir
uma pessoa que não seja um funcionário, indique isso por meio de valores nulos */

SELECT P.CPF, P.Nome, P.Sexo, P.`E-mail`, P.Telefone, P.idPessoa, P.tipoPessoa,
       F.`Carga Horaria`, F.Salario, F.TipoFunc, F.`Data Inicio`, F.Departamento_idDepto

FROM pessoa P LEFT OUTER JOIN funcionario F ON P.CPF = F.`CPF Func`
ORDER BY P.Nome;
```

```sql
/* Consulta 07: Recupere o nome e o cpf dos funcionários que não contêm produtos em estoque

SELECT funcionario.`CPF Func`, pessoa.Nome
FROM funcionario JOIN pessoa ON funcionario.`CPF Func` = pessoa.CPF
WHERE funcionario.`CPF Func` NOT IN (SELECT `Funcionario_CPF Func`
                                     FROM `produtos em estoque_has_funcionario`)
ORDER BY pessoa.Nome;
```

```sql
/* Consulta 08: Recupere o nome e o salario dos funcionarios que trabalham no departamento
   "Vendas" OU daqueles que possuem produtos em estoque */

SELECT funcionario.`CPF Func`, pessoa.Nome
FROM funcionario JOIN pessoa ON funcionario.`CPF Func` = pessoa.CPF
     JOIN departamento ON funcionario.Departamento_idDepto = departamento.idDepto

WHERE departamento.`Nome Depto` = "Vendas"
UNION
SELECT funcionario.`CPF Func`, pessoa.Nome
FROM funcionario JOIN pessoa ON funcionario.`CPF Func` = pessoa.CPF
     JOIN `produtos em estoque_has_funcionario` P ON P.`Funcionario_CPF Func` = pessoa.CPF;
```

```sql
/* Consulta 09: Recupere o nome e o salario das funcionarias do sexo feminino que
   ganham a mesma quantidade ou mais que os funcionarios do sexo masculino */

SELECT pessoa.Nome, funcionario.Salario
FROM funcionario JOIN pessoa ON funcionario.`CPF Func` = pessoa.CPF
WHERE pessoa.Sexo = 'F' AND
funcionario.Salario >= SOME (SELECT funcionario.Salario
                             FROM funcionario JOIN pessoa ON funcionario.`CPF Func` = pessoa.CPF
                             WHERE pessoa.Sexo = 'M' AND pessoa.tipoPessoa = 'f');
```

```sql
/* Consulta 10: Recupero o nome e o salario dos funcionarios que: tenham o nome que se
inicie com a letra F e que ganhem mais que todos os funcionarios que trabalham no departamento 2 */

SELECT pessoa.Nome, funcionario.Salario
FROM funcionario JOIN pessoa ON funcionario.`CPF Func` = pessoa.CPF
WHERE pessoa.Nome LIKE 'f%' AND funcionario.Salario > ALL (SELECT funcionario.Salario
                                                           FROM funcionario);
```

```sql
/* Consulta 11: */ -- IS NULL

/*O operador IS NULL é usado para verificar se um valor em uma coluna é nulo.
Ele retorna verdadeiro se a coluna for nula e falso se a coluna tiver um valor*/

/*Suponha que você queira encontrar todos os funcionários que
não possuem um departamento atribuído (ou seja, o campo
Departamento_idDepto na tabela "Funcionario" é nulo).

A consulta seria assim: */

SELECT *
FROM Funcionario f
WHERE f.Departamento_idDepto IS NULL;
```

```sql
/* Consulta 12: */ -- EXISTS

/*Esta consulta seleciona os nomes dos departamentos da tabela
Departamento onde existe pelo menos um registro na tabela
Funcionário com o mesmo Departamento_idDepto.

O operador EXISTS é utilizado para verificar a
existência de tais registros na subconsulta.*/

SELECT d.`Nome Depto`
FROM `mydb`.`Departamento` d
WHERE EXISTS (
  SELECT 1
  FROM `mydb`.`Funcionario` f
  WHERE f.`Departamento_idDepto` = d.`idDepto`
);
```

```sql
/*Consulta 13*/ -- Ok

/*Encontre o nome, e-mail e telefone do departamento que tem o maior número de funcionários.*/

SELECT d.`Nome Depto`, d.`E-mail`, td.`Telefone Depto`
FROM `Departamento` d
INNER JOIN `Telefone Depto` td ON d.`idDepto` = td.`Departamento_idDepto`
WHERE d.`idDepto` = (SELECT `Departamento_idDepto`
                     FROM `Funcionario`
                     GROUP BY `Departamento_idDepto`
                     ORDER BY COUNT(*) DESC
                     LIMIT 1);
```

```sql
/*Consulta 14*/

/*Encontre o número de clientes atendidos por cada atendente.*/

SELECT a.`CPF Aten`, COUNT(c.`Id Cliente`) AS 'Num Clientes Atendidos'
FROM `mydb`.`Atendente` a
LEFT JOIN `mydb`.`Cliente` c ON a.`Id Cliente` = c.`Id Cliente`
GROUP BY a.`CPF Aten`;
```

g)

```sql
/* view_cliente_cpf_nome: Esta view mostra o nome e CPF dos clientes */

/* view1 */

CREATE VIEW view_cliente_cpf_nome AS
SELECT pessoa.Nome, cliente.`CPF Cliente`
FROM `mydb`.cliente, `mydb`.pessoa;
/*Para usar esta view, basta executar o seguinte comando:*/
SELECT * FROM view_cliente_cpf_nome;
```

```sql
/*view2*/

/*concatena as colunas Rua e Num Cliente, Cidade Cliente, Estado Cliente e CEP
em uma única coluna chamada Endereco, separando os valores com vírgulas e espaços.
Ou seja, a view apresenta os endereços completos de todos os clientes na tabela Cliente.*/

CREATE VIEW view_cliente_endereco AS
SELECT `Id Cliente`, CONCAT(`Rua e Num Cliente`, ', ', `Cidade Cliente`,
                            ', ', `Estado Cliente`, ' ', CEP) AS Endereco
FROM `mydb`.`Cliente`;

SELECT * FROM view_cliente_endereco;
```

```sql
/*view3*/

/*seleciona as colunas tipoPessoa e Sexo da tabela Pessoa no banco de dados mydb.
Ela é criada com a intenção de facilitar a consulta desses dados, agrupando-os em
uma única view para que possam ser consultados de forma mais simples e organizada.*/

CREATE VIEW view_pessoa_tipo_sexo AS
SELECT tipoPessoa, Sexo
FROM `mydb`.`Pessoa`;
SELECT * FROM view_pessoa_tipo_sexo;
```

h)

```sql
CREATE USER 'usuario1'@'localhost' IDENTIFIED BY 'senha1';
GRANT ALL PRIVILEGES ON `mydb`.* TO 'usuario1'@'localhost';
CREATE USER 'usuario2'@'localhost' IDENTIFIED BY 'senha2';
GRANT SELECT, INSERT, UPDATE ON mydb.cliente TO 'usuario2'@'localhost';

REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'usuario1'@'localhost';
REVOKE SELECT, INSERT, UPDATE ON mydb.cliente FROM 'usuario2'@'localhost';
```

i)

```sql
-- configuração para habilitar funções deterministicas
SET GLOBAL log_bin_trust_function_creators = 1;


-- FUNCAO 1 ---------------------------------------------------

-- Delete a função se ela existir
DROP FUNCTION IF EXISTS calcula_bonus;

DELIMITER //

/* Essa função recebe dois parametros: carga horaria e salario,
e calcula um bonus baseado nesses valores,se a carga horaria
for maior que 40 o bonus é de 10%, senão o bonus é de 5% */

CREATE FUNCTION calcula_bonus(`Carga Horaria` INT, salario INT) RETURNS INT
BEGIN
  DECLARE bonus INT;

  IF `Carga Horaria` > 40 THEN
    SET bonus = salario * 0.1;
  ELSE
    SET bonus = salario * 0.05;
  END IF;

  RETURN bonus;
END //
```

```sql
DELIMITER ;

-- Exemplos

/* chamando a função passando como parametro a carga horaria,
salario e bonus e retornando a seleção de idDepto, idPessoa
Carga horaria, salario e bonus de todos os funcionarios */
SELECT `Departamento_idDepto`, Pessoa_idPessoa, `Carga Horaria`, salario,
                              calcula_bonus(`Carga Horaria`, salario) AS bonus_total
FROM Funcionario;


/* selecionando o idDepto, Carga horaria, salario e bonus de todos
   os funcionarios de um departamento espeficicado pelo idDepto */
SELECT `Departamento_idDepto`, Pessoa_idPessoa, `Carga Horaria`, salario,
                              calcula_bonus(`Carga Horaria`, salario) AS bonus_total
FROM Funcionario
WHERE `Departamento_idDepto` = 1;


/* outro exemplo de uso, some o bonus total de cada departamento
baseado na soma dos bonus de todos os funcionarios de cada departamento */

SELECT d.`Nome Depto`, idDepto, SUM(calcula_bonus(f.`Carga Horaria`,
                                    f.salario)) AS bonus_Depto
FROM Funcionario f
JOIN Departamento d ON f.`Departamento_idDepto` = d.idDepto
GROUP BY d.`Nome Depto`;

 -- FUNCAO 2 ------------------------------------------------

 -- Delete a função se ela existir
 DROP FUNCTION IF EXISTS salario_total_Depto;

 DELIMITER //

/* Esta funcao soma o salario e o bonus de todos os funcionarios de cada
departamento para isso é chamada a funcao calcula_bonus que já foi criada */

 CREATE FUNCTION salario_total_Depto(idDepto INT) RETURNS DECIMAL(10, 2)
 BEGIN
   DECLARE salario_Total DECIMAL(10, 2);
   DECLARE bonus DECIMAL(10, 2);

   SELECT SUM(Salario) INTO salario_Total
   FROM `mydb`.Funcionario WHERE idDepto = `Departamento_idDepto`;

   SELECT SUM(calcula_bonus(`Carga Horaria`, Salario)) INTO bonus
   FROM Funcionario WHERE idDepto = `Departamento_idDepto`;

   SET salario_Total = salario_Total + bonus;

   RETURN salario_Total;
 END //

 DELIMITER ;
```

```sql
-- Exemplos

/* chamando a funcao passando como parametro o idDepto desejado
e retornando uma seleção do Nome Depto, idDepto e o resultado
da função que é a soma total do salario do departamento */

SELECT `Nome Depto`, idDepto, salario_total_Depto(1) AS `salario + bonus`
FROM Departamento
WHERE idDepto = 1 ;

SELECT `Nome Depto`, idDepto, salario_total_Depto(2) AS `salario + bonus`
FROM Departamento
WHERE idDepto = 2;

SELECT `Nome Depto`, idDepto, salario_total_Depto(3) AS `salario + bonus`
FROM Departamento
WHERE idDepto = 3;



-- FUNCAO 3 ------------------------------------------------

-- Delete a função se ela existir
DROP FUNCTION IF EXISTS calcula_anos_trabalhados;

DELIMITER //

/* Esta função calcula os anos trabalhados de cada funcionario,
recebendo como parametro a Data Inicio de cada funcionario */

CREATE FUNCTION calcula_anos_trabalhados(`Data Inicio` DATE) RETURNS INT
BEGIN
  DECLARE anos_trabalhados INT;

  SET anos_trabalhados = YEAR(CURRENT_DATE()) - YEAR(`Data Inicio`);

  IF MONTH(CURRENT_DATE()) < MONTH(`Data Inicio`) THEN
    SET anos_trabalhados = anos_trabalhados - 1;

  ELSEIF MONTH(CURRENT_DATE()) = MONTH(`Data Inicio`)
                            AND DAY(CURRENT_DATE()) < DAY(`Data Inicio`) THEN

    SET anos_trabalhados = anos_trabalhados - 1;
  END IF;

  RETURN anos_trabalhados;
END //

DELIMITER ;
```

```sql
-- Exemplo

/* chamando a funcao passando como parametro a Data Inicio de cada funcionario e retornando
uma seleção do nome, data inicio e anos trabalhados de cada funcionario */

SELECT `Departamento_idDepto`,Pessoa_idPessoa, `Data Inicio`,
        calcula_anos_trabalhados(`Data Inicio`) AS anos_trabalhados

FROM Funcionario;
```

j)

```sql
/*Este trigger é acionado sempre que um novo registro é inserido na tabela "Funcionario".
Ele simplesmente exibe uma mensagem de log na console do MySQL com informações sobre o novo registro inserido.*/
/*para ver as mensagens no console, basta executar select * from `mydb`.`log`;*/

DELIMITER $$
CREATE TRIGGER `insert_funcionario` AFTER INSERT ON `Funcionario`
FOR EACH ROW
BEGIN
SELECT CONCAT('Novo registro inserido na tabela Funcionario: CPF Func - ', NEW.`CPF Func`, ', Departamento_idDepto - ',
NEW.`Departamento_idDepto`, ', Pessoa_idPessoa - ', NEW.`Pessoa_idPessoa`) AS `msg` INTO @msg;

INSERT INTO `log` (`mensagem`) VALUES (@msg);
END
$$
delimiter ;

/*Trigger de alteração*/
/*Este trigger é acionado sempre que um registro existente é atualizado na tabela "Pessoa".
Ele atualiza a tabela "Cliente" com o novo valor do campo "E-mail" do registro atualizado na tabela "Pessoa".*/

delimiter $$
CREATE TRIGGER `update_cliente` AFTER UPDATE ON `Pessoa`
FOR EACH ROW
BEGIN
  UPDATE `Cliente` SET `CPF Cliente` = NEW.`CPF` WHERE `Id Cliente` = OLD.`idPessoa`; -- Trigger está funcionando.
END
$$
delimiter ;
```

```sql
/* Trigger de exclusão

Este trigger é acionado sempre que um registro é excluído na tabela "Departamento".
Ele exclui todos os registros correspondentes na tabela "Telefone Depto".*/

delimiter $$

CREATE TRIGGER `delete_departamento` BEFORE DELETE ON `Departamento`
FOR EACH ROW
BEGIN
  DELETE FROM `Telefone Depto` WHERE `Departamento_idDepto` = OLD.`idDepto`;
END
$$

delimiter ;
```