UNIESP – Centro Universitário

SISTEMAS DE INFORMAÇÃO/SISTEMAS PARA INTERNET

Gabriel de Lima Arruda
Vinicius Nascimento Cruz
Iremar Flor de Souza Filho
Lucas Gabriel Silva de Azevedo
Ícaro Raphael Queiroz Cavalcante
Luiz Humberto Saldanha Gomes de Souza

Proposta de um modelo de banco de dados comercial na área de supermercado

Cabedelo
2022

Gabriel de Lima Arruda
Vinicius Nascimento Cruz
Iremar Flor de Souza Filho
Lucas Gabriel Silva de Azevedo
Ícaro Raphael Queiroz Cavalcante
Luiz Humberto Saldanha Gomes de Souza

Proposta de um modelo de banco de dados comercial na área de supermercado

Projeto apresentado na disciplina de Banco de Dados I, no curso de Sistemas de Informação/Sistemas para Internet, sob orientação do Professor Ms. Fábio Nicácio de Medeiros.

Cabedelo
2022

SUMÁRIO

1. MODELO DESCRITIVO

O Supermercado Bem-te-vi é um comércio de médio porte e deseja elaborar um sistema de informação para controlar suas atividades. O objetivo é modelar uma solução que atenda as principais áreas da empresa. Os requisitos estão listados abaixo:

1 - Todos os produtos devem ser cadastrados para que haja um completo controle de estoque, valor de custo, valor de venda, código do produto, categoria a qual pertence, uma breve descrição e o nome. Todos esses dados são importantes para a manutenção do estoque, controle de gastos e facilidade na declaração de imposto;

2 - Nos registros das vendas devem conter os produtos presente no carrinho de compras, o valor da venda, o funcionário que realizou a venda, o método utilizado para o pagamento e o cliente que realizou a compra, todos esses dados são de extrema importância para o setor financeiro do supermercado;

3 - O cadastro do cliente precisa ser otimizado, ou seja, rápido e eficiênte, e no cadastro é necessário informar apenas o nome e o CPF do cliente, são dados importantes para a parte financeira do supermercado, além de ser importante para o setor do marketing, pois é necessário saber se os clientes estão retornando ou pararam de frequentar o local;

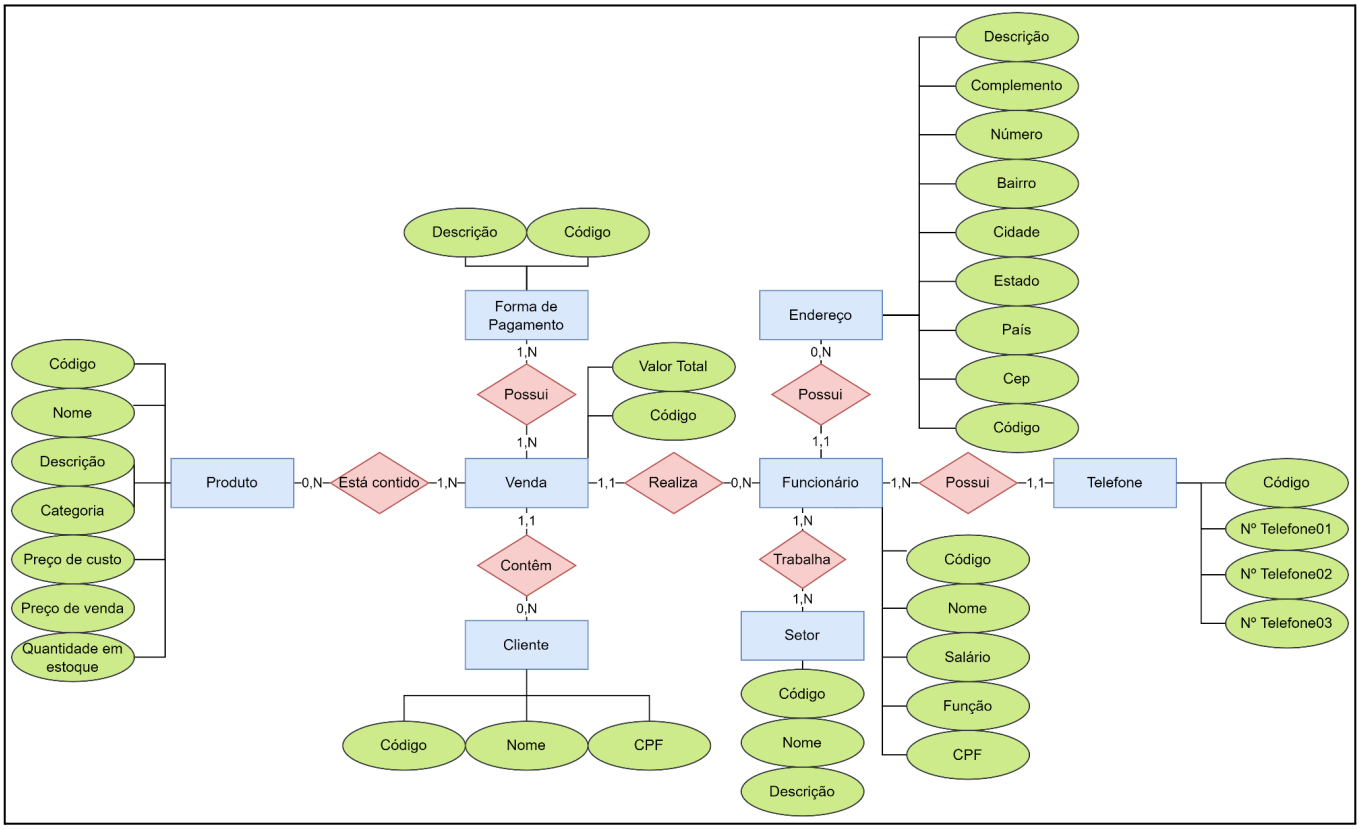
4 - No supermercado também é de grande importância o cadastro dos setores a qual os funcionários trabalham, deverá ser um cadastro simples e bem resumido contendo apenas o nome e a descrição do setor, lembrando que deverá existir um setor chamado “Setor geral” para os funcionários que não possuem setor fixo;

5 - Na parte do cadastro dos funcionários já é mais complexo, visto que são todas as peças que compõem o supermercado e fazem ele funcionar perfeitamente, para isso é necessário informar o nome, salário, CPF, endereço, telefones, setor em que trabalha e por último a função exercida pelo funcionário, pois poderá ser vendedor, zelador, estoquista, contador e gerente;

6 - No registro dos endereços dos funcionários é importante informar o CEP de cada funcionário, assim como país, o estado, a cidade, o bairro, o número da residência, o complemento caso o funcionário morar em condomínio, e a descrição onde vai conter pelo menos um ponto de referência;

7 - Já por fim, a respeito do registro de telefone do funcionário deverá ser da seguinte forma, o funcionário deverá ter pelo menos um telefone cadastrado, podendo ter até três telefones no registro, e deverá também ser registrado o DDD da região a qual pertence cada telefone;

2. MODELO CONCEITUAL



Fonte: https://drive.google.com/file/d/1DpbgTxG1HOeRfLF9-06d-NVB7Hk__Wa4/view?usp=sharing

3. EXPLICAÇÃO DO MODELO CONCEITUAL

A partir do modelo descritivo, cria-se o modelo conceitual composto com as seguintes entidades e atributos:

- Produto: Composto pelos atributos código, nome, descrição, categoria, preço de custo, preço de venda e quantidade em estoque. Tem como objetivo armazenar os dados de todos os produtos do supermercado;

- Forma de pagamento: Composto pelos atributos código e descrição, tem como objetivo armazenar a forma de pagamento realizada na venda;

- Cliente: Composto pelos atributos código, nome e CPF, tem como objetivo armazenar os dados de todos os clientes que realizaram compra no supermercado;

- Venda: Composto pelos atributos código e valor total, é a entidade principal do banco, pois a venda é o pilar do supermercado, e tem como objetivo armazenar todas as vendas do supermercado;

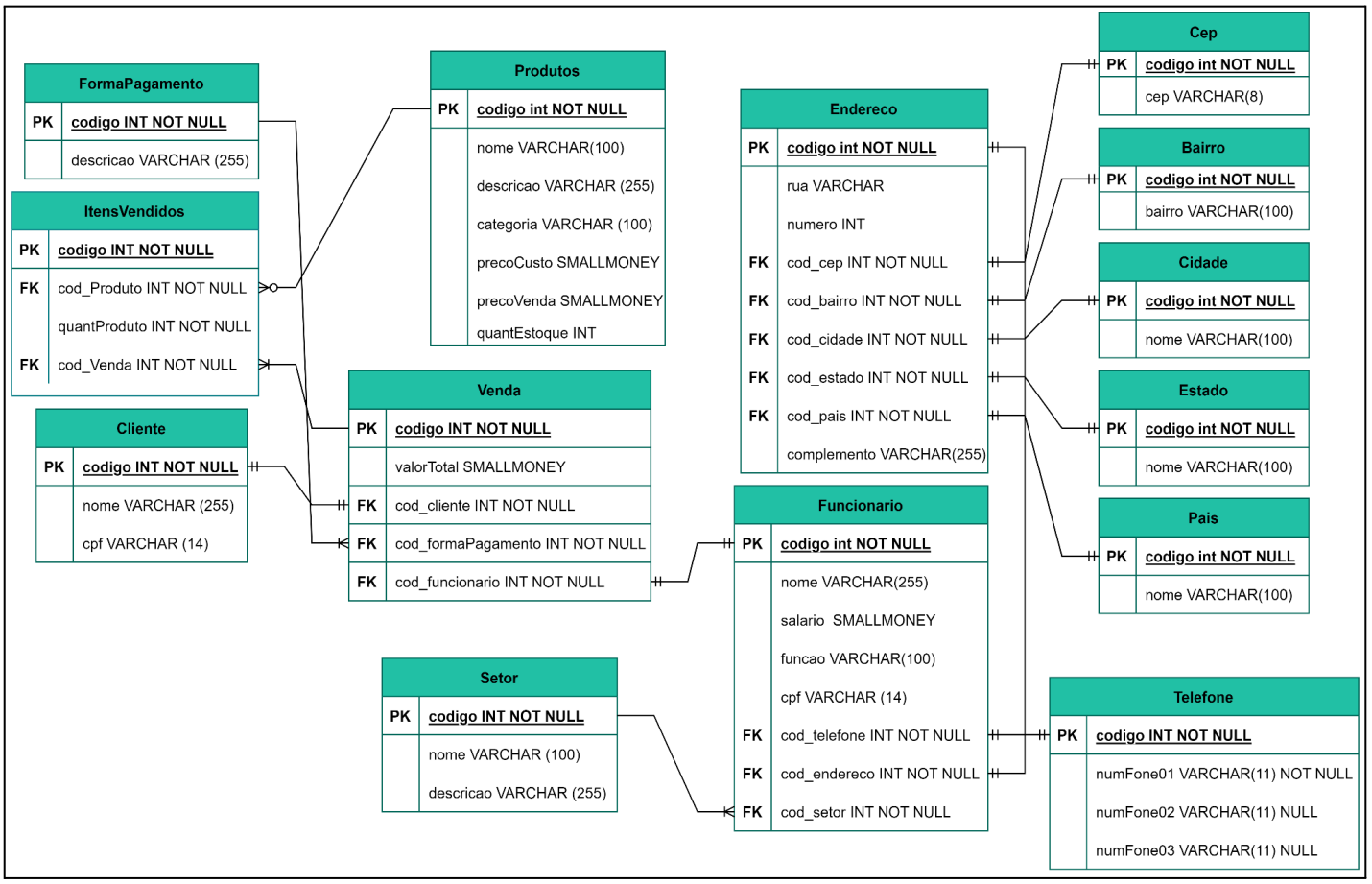
- Funcionário: Composto pelos atributos código, nome, salário, função e CPF, tem como objetivo armazenar os dados de todos os funcionários que trabalham no supermercado;

- Setor: Composto pelos atributos código, nome e descrição, tem como objetivo armazenar todos os setores presente no supermercado;

- Telefone: Composto pelos atributos código, número de telefone 1, número de telefone 2 e número de telefone 3, tem como objetivo armazenar os números de telefone dos funcionários, onde o mínimo aceito pelo sistema é de um telefone, e suporta cadastrar até três telefones;

- Endereço: Composto pelos atributos código, cep, país, estado, cidade, bairro, número, complemento e descrição, tem como objetivo armazenar o endereço de todos os funcionários que foram contratados pela empresa.

4. MODELO LÓGICO



FONTE: https://drive.google.com/file/d/1h8pZuNK6COQp4crdDBQQO0RUBUe8dnyj/view?usp=sharing

5. EXPLICAÇÃO DO MODELO LÓGICO

O modelo lógico é uma das etapas mais importantes e também é onde o banco de dados começa a tomar forma, pois é no modelo lógico que começa a se desenvolver tabelas baseadas nas etapas anteriores, e também se começa a inserir os tipos dos dados. Segue a explicação de cada tabela abaixo:

- **Tabela "Cliente":** A tabela cliente possui três atributos, o código que é do tipo inteiro não nulo, e também é a chave primária da tabela, o nome que é do tipo varchar e pode possuir até 255 caracteres, e o CPF que é do tipo varchar e possui 14 caracteres, o "CPF" será inserido no formato "000.000.000-00";

- **Tabela "Forma de pagamento":** Essa tabela é composta por dois atributos, o código que é do tipo inteiro não nulo, e a descrição, onde será informado e descrito os métodos de pagamento;

- **Tabela "Cep":** Composta por dois campos, o código do tipo inteiro não nulo, que também é a chave primária, e o CEP no tipo varchar de 8 caracteres e que será inserido no formato "00000000" com todos os números juntos;

- **Tabela "Bairro":** Composta por dois campos, a chave primária identificada como código do tipo inteiro não nulo, e o bairro, onde será inserido o nome do bairro;

- **Tabela "Cidade":** Possui dois campos, o código do tipo inteiro não nulo e o campo cidade, do tipo varchar podendo ter até 100 caracteres;

- **Tabela "Estado":** Com dois campos, o código, como chave primária do tipo inteira não nula, e o estado, do tipo varchar podendo ter até 100 caracteres;

- **Tabela "País":** Composto pelos campos código e país, onde "código" é a chave primária do tipo inteira não nula, e "país" do tipo varchar;

- **Tabela "Endereço":** Tem em sua composição os atributos código, rua, numero, cod_cep, cod_bairro, cod_cidade, cod_estado, cod_pais, complemento, onde o "código" é a chave primária, e os atributos que são antecedidos por "cod" são chaves estrangeiras que referenciam outras tabelas;

- **Tabela "Telefone":** Possui os campos código, numFone01, numFone02 e numFone03, onde o "código" é a chave primária, e apenas o "numFone01" que é o campo de telefone obrigatório;

- **Tabela "Setor":** Tem três campos, código como a chave primária, nome para inserção do nome do setor, e descrição para inserir informações importantes acerca do setor;

- **Tabela “Itens vendidos”:** Na tabela “Itens Vendidos” está presente quatro atributos, são eles: código, cod_produto, quantProduto, cod_venda. O “código” é a chave primária da tabela do tipo inteiro não nulo, o “cod_produto” faz referência a tabela produto, o “quantProduto” é a quantidade do produto vendido, e o “cod_venda” faz referência a tabela venda.

- **Tabela “Produto”:** Composto pelos atributos código, nome, descrição, categoria, preçoCusto, preçoVenda e quantEstoque, tem como chave primária o “código” sendo do tipo inteiro não nulo. Os atributos “preçoCusto” e “preçoVenda” são do tipo smallmoney, “quantEstoque” é do tipo inteiro, e os demais atributos são do tipo varchar;

- **Tabela “Venda”:** Essa tabela é composta pelo código, valorTotal, cod_cliente, cod_formaPagamento e cod_funcionario, onde o “código” é a chave primária do tipo inteira não nula, o “valorTotal” é do tipo smallmoney, e os demais atributos com o prefixo “cod” fazem referência a outras tabelas;

- **Tabela “Funcionario”:** Tem em sua composição os atributos código, nome, salário, função, cpf, cod_telefone, cod_endereço e cod_setor, onde o “código” é a chave primária do tipo inteira não nula, e também “nome”, “função” e “cpf” são do tipo varchar, além dos atributos com prefixo “cod” que fazem referência a outras tabelas;

6. DICIONÁRIO DE DADOS

TABELA: Produto							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código do produto		X		
nome	varchar(100)	não	Nome do produto				
descricao	varchar(255)	não	Descrição do produto				
categoria	varchar(100)	não	Categoria do produto				
precoCusto	smallmoney	não	Preço de custo do produto				
precoVenda	smallmoney	não	Preço de venda do produto				
quantEstoque	int	não	Quantidade em estoque do produto				

TABELA: Cliente							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código do cliente		X		
nome	varchar(255)	não	Nome do cliente				
cpf	varchar(14)	não	CPF do cliente				X

TABELA: FormaPagamento							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código da forma de pagamento		X		
descricao	varchar(255)	não	Descrição da forma de pagamento				

TABELA: Setor							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código do setor		X		
nome	varchar(100)	não	Nome do setor				
descricao	varchar(255)	não	Descrição do setor				

TABELA: Telefone							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código do telefone		X		
numFone01	varchar(11)	não	Primeira opção de telefone				
numFone02	varchar(11)	sim	Segunda opção de telefone				
numFone03	varchar(11)	sim	Terceira opção de telefone				

TABELA: Cep							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código do CEP		X		
cep	varchar(8)	não	CEP no formato “00000000”				X

TABELA: Bairro							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codig	int	não	Código do bairro		X		
nome	varchar(100)	não	Nome do bairro				

TABELA: Cidade							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código da cidade		X		
nome	varchar(100)	não	Nome da cidade				

TABELA: Estado							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código do estado		X		
nome	varchar(100)	não	Nome do estado				

TABELA: Pais							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código do do país		X		
nome	varchar(100)	não	Nome do país				

TABELA: Endereco							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código do endereço		X		
rua	varchar	não	Nome da rua				
numero	int	não	Número da residência				
cep	int	não	Código da tabela Cep	Tabela Cep		X	
bairro	Int	não	Código da tabela Bairro	Tabela Bairro		X	
cidade	int	não	Código da tabela Cidade	Tabela Cidade		X	
estado	int	não	Código da tabela Estado	Tabela Estado		X	
pais	int	não	Código da tabela Pais	Tabela Pais		X	
complemeto	varchar(255)	sim	Complemento do endereço				

TABELA: Funcionario							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código do funcionário		X		
nome	varchar(255)	não	Nome do funcionário				
salario	smallmoney	não	Salário do funcionário				
funcao	varchar(100)	não	Função do funcionário				
cpf	varchar(14)	não	CPF no formato “000.000.000-00”				X
telefone	int	não	Código da tabela Telefone	Tabela Telefone		X	
endereco	int	não	Código da tabela Endereco	Tabela Endereco		X	
setor	int	não	Código da tabela Setor	Tabela Setor		X	

TABELA: Venda							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código da venda		X		
valorTotal	smallmoney	não	Valor total da venda				
cliente	int	não	Código da tabela Cliente	Tabela Cliente		X	
formaPagamento	int	não	Código da tabela formaPagamento	Tabela FormaPagamento		X	
funcionario	int	não	Código da tabela Funcionario	Tabela Funcionario		X	

TABELA: ItensVendidos							
ATRIBUTOS	TIPO	NULO	DESCRIÇÃO	DOMÍNIO	CHAVE		
					PRI	EST	CAN
codigo	int	não	Código dos itens vendidos		X		
produto	int	não	Código do Produto	Tabela Produto		X	
quantProduto	int	não	Quantidade de produto na venda				
venda	int	não	Código da venda	Tabela Venda		X	

7. RESTRIÇÕES

TABELA: Produto			
COLUNA	**TIPO**	**NOME**	**EXPRESSÃO**
codigo	Chave primária	pk_produto	primary key

TABELA: Cliente			
COLUNA	**TIPO**	**NOME**	**EXPRESSÃO**
codigo	Chave primária	pk_cliente	primary key
cpf	Chave candidata	un_cliente_cpf	unique

TABELA: FormaPagamento			
COLUNA	**TIPO**	**NOME**	**EXPRESSÃO**
codigo	Chave primária	pk_formaPagamento	primary key

TABELA: Setor			
COLUNA	**TIPO**	**NOME**	**EXPRESSÃO**
codigo	Chave primária	pk_setor	primary key

TABELA: Telefone			
COLUNA	**TIPO**	**NOME**	**EXPRESSÃO**
codigo	Chave primária	pk_telefone	primary key

TABELA: Cep			
COLUNA	**TIPO**	**NOME**	**EXPRESSÃO**
codigo	Chave primária	pk_cep	primary key
cep	Chave candidata	un_cep_cep	unique

TABELA: Bairro			
COLUNA	TIPO	NOME	EXPRESSÃO
codigo	Chave primária	pk_bairro	primary key

TABELA: Cidade			
COLUNA	TIPO	NOME	EXPRESSÃO
codigo	Chave primária	pk_cidade	primary key

TABELA: Estado			
COLUNA	TIPO	NOME	EXPRESSÃO
codigo	Chave primária	pk_estado	primary key

TABELA: Pais			
COLUNA	TIPO	NOME	EXPRESSÃO
codigo	Chave primária	pk_pais	primary key

TABELA: Endereco			
COLUNA	TIPO	NOME	EXPRESSÃO
codigo	Chave primária	pk_endereco	primary key
cep	Chave estrangeira que referencia a tabela Cep	fk_endereco_cep	foreign key references Cep(codigo)
bairro	Chave estrangeira que referencia a tabela Bairro	fk_endereco_bairro	foreign key references Bairro(codigo)
cidade	Chave estrangeira que referencia a tabela Cidade	fk_endereco_cidade	foreign key references Cidade(codigo)
estado	Chave estrangeira que referencia a tabela Estado	fk_endereco_estado	foreign key references Estado(codigo)
pais	Chave estrangeira que referencia a tabela Pais	fk_endereco_pais	foreign key references Pais(codigo)

TABELA: Funcionario			
COLUNA	**TIPO**	**NOME**	**EXPRESSÃO**
codigo	Chave primária	pk_funcionario	primary key
cpf	Chave candidata	un_funcionario_cpf	unique
telefone	Chave estrangeira que referencia a tabela Telefone	fk_funcionario_telefone	foreign key references Telefone(codigo)
endereco	Chave estrangeira que referencia a tabela Endereco	fk_funcionario_endereco	foreign key references Endereco(codigo)
setor	Chave estrangeira que referencia a tabela Setor	fk_funcionario_setor	foreign key references Setor(codigo)

TABELA: Venda			
COLUNA	**TIPO**	**NOME**	**EXPRESSÃO**
codigo	Chave primária	pk_venda	primary key
cliente	Chave estrangeira que referencia a tabela Cliente	fk_venda_cliente	foreign key references Cliente(codigo)
formaPagamento	Chave estrangeira que referencia a tabela formaPagamento	fk_venda_formaPagamento	foreign key references FormaPagamento(codigo)
funcionario	Chave estrangeira que referencia a tabela Funcionario	fk_venda_funcionario	foreign key references Funcionario(codigo)

TABELA: ItensVendidos			
COLUNA	**TIPO**	**NOME**	**EXPRESSÃO**
codigo	Chave primária	pk_itensVendidos	primary key
produto	Chave estrangeira que referencia a tabela Produto	fk_itensVendidos_produto	foreign key references Produto(codigo)
venda	Chave estrangeira que referencia a tabela Venda	fk_itensVendidos_venda	foreign key references Venda(codigo)

8. EXPLICAÇÃO DO DICIONÁRIO DE DADOS

O dicionário de dados é composto por duas partes, sendo elas:

- **Primeira parte:** Consiste de tabelas que mostram de forma geral cada tabela do banco, tem em sua composição os seguintes dados:

    - Atributo: Nada mais é do que as colunas das tabelas do banco escritas da mesma maneira que estão no banco de dados;

    - Tipo: É o tipo de dado que será inserido no atributo pertencente ao banco de dados;

    - Nulo: É a coluna que mostra se o respectivo atributo pode ser “Nulo” ou “Não nulo” dentro do banco;

    - Descrição: É um breve resumo falando do que se trata os respectivos atributos dos bancos;

    - Domínio: Essa coluna só irá ser preenchida caso o respectivo atributo possua domínio, um exemplo é se o atributo for chave estrangeira, então o domínio será a referência da tabela;

    - Chave: Essa coluna é composta pela chave primária (PRI), estrangeira (EST) e candidata (CAN), e só será marcada caso o respectivo atributo for uma chave;

- **Segunda parte:** Consiste nas restrições de cada tabela, e tem em sua composição os seguintes dados:

    - Coluna: É o campo onde será inserido o nome da coluna que possui a determinada restrição;

    - Tipo: É o tipo da restrição da respectiva coluna a qual pertence;
    - Nome: É a nomenclatura que será utilizada dentro do banco de dados para referenciar a restrição;

    - Expressão: É a expressão utilizada dentro do banco de dados para definir a restrição da respectiva coluna;

9. SCRIPT DE CRIAÇÃO DO BANCO

```
create database Supermercado
on (name=supermercado_dat,
        filename = 'C:\SQLSV\supermercado_bd.mdf')
go

use Supermercado
go

create table Produto(
        codigo int not null
                constraint pk_produto primary key,
        nome varchar(100) not null,
        descricao varchar(255) not null,
        categoria varchar(100) not null,
        precoCusto smallmoney not null,
        precoVenda smallmoney not null,
        quantEstoque int not null
)

create table Cliente(
        codigo int not null
                constraint pk_cliente primary key,
        nome varchar(255) not null,
        cpf varchar(14) not null
                constraint un_cliente_cpf unique
)

create table FormaPagamento(
        codigo int not null
                constraint pk_formaPagamento primary key,
        descricao varchar(255) not null
)

create table Setor(
        codigo int not null
                constraint pk_setor primary key,
        nome varchar(100) not null,
        descricao varchar(255) not null
)

create table Telefone(
        codigo int not null
                constraint pk_telefone primary key,
        numFone01 varchar(11) not null,
        numFone02 varchar(11) null,
        numFone03 varchar(11) null
)

create table Cep(
        codigo int not null
                constraint pk_cep primary key,
        cep varchar(8) not null
                constraint un_cep_cep unique
)
```

```
create table Bairro(
	codigo int not null
		constraint pk_bairro primary key,
	nome varchar(100) not null
)

create table Cidade(
	codigo int not null
		constraint pk_cidade primary key,
	nome varchar(100) not null
)

create table Estado(
	codigo int not null
		constraint pk_estado primary key,
	nome varchar(100) not null
)

create table Pais(
	codigo int not null
		constraint pk_pais primary key,
	nome varchar(100) not null
)

create table Endereco(
	codigo int not null
		constraint pk_endereco primary key,
	rua varchar(255) not null,
	numero int not null,
	cep int not null
		constraint fk_endereco_cep foreign key references Cep(codigo),
	bairro int not null
		constraint fk_endereco_bairro foreign key references Bairro(codigo),
	cidade int not null
		constraint fk_endereco_cidade foreign key references Cidade(codigo),
	estado int not null
		constraint fk_endereco_estado foreign key references Estado(codigo),
	pais int not null
		constraint fk_endereco_pais foreign key references Pais(codigo),
	complemento varchar(255) null
)

create table Funcionario(
	codigo int not null
		constraint pk_funcionario primary key,
	nome varchar(255) not null,
	salario smallmoney not null,
	funcao varchar(100) not null,
	cpf varchar(14) not null
		constraint un_funcionario_cpf unique,
	telefone int not null
		constraint fk_funcionario_telefone foreign key references Telefone(codigo),
	endereco int not null
		constraint fk_funcionario_endereco foreign key references Endereco(codigo),
	setor int not null
		constraint fk_funcionario_setor foreign key references Setor(codigo)
)
```

```
create table Venda(
	codigo int not null
		constraint pk_venda primary key,
	valorTotal smallmoney not null,
	cliente int not null
		constraint fk_venda_cliente foreign key references Cliente(codigo),
	formaPagamento int not null
		constraint fk_venda_formaPagamento foreign key references
		FormaPagamento(codigo),
	funcionario int not null
		constraint fk_venda_funcionario foreign key references Funcionario(Codigo)
)

create table ItensVendidos(
	codigo int not null
		constraint pk_itensVendidos primary key,
	produto int not null
		constraint fk_itensVendidos_produto foreign key references Produto(codigo),
	quantProduto int not null,
	venda int not null
		constraint fk_itensVendidos_venda foreign key references Venda(codigo)
)
```

10. SCRIPT DE INSERÇÃO DE DADOS

```
use Supermercado
go

-- TABELA PRODUTO
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000001, 'Arroz Mundial', 'Arroz da marca Mundial', 'Alimento', 22.90, 27.70, 46);
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000002, 'Feijão Mundial', 'Feijão da marca Mundial', 'Alimento', 19.70, 23.80, 67);
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000003, 'Hambúrguer Seara', 'Hambúrguer da marca Seara', 'Alimento', 9.99, 12.99,
102);
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000004, 'Leite Guarabira', 'Leite da marca Guarabira', 'Alimento', 15.23, 17.49, 39);
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000005, 'Guaraná Jesus', 'Refrigerante guaraná da marca Jesus', 'Alimento', 5.99,
8.99, 208);
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000006, 'Chocolate Garoto', 'Barra de chocolate da marca garoto', 'Alimento', 4.99,
7.99, 137);
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000007, 'Macarrão MassaDaBoa', 'Macarrão da marca MassaDaBoa', 'Alimento', 4.78,
6.90, 112);
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000008, 'Detergente LimpaGeral', 'Detergente da marca LimpaGeral', 'Material de
limpeza', 3.99, 5.49, 58);
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000009, 'Sabonete Senador', 'Sabonete da marca Senador', 'Higiene pessoal', 0.99,
1.99, 1023);
insert dbo.Produto (codigo, nome, descricao, categoria, precoCusto, precoVenda,
quantEstoque)
values(000010, 'Sabão em pó Omo', 'Sabão em pó da marca Omo', 'Material de limpeza',
6.90, 8.90, 78);
```

```
-- TABELA CLIENTE
Insert dbo.Cliente (codigo, nome, cpf)
values(000001, 'Alexandre','132.752.294-17');
Insert dbo.Cliente (codigo, nome, cpf)
values(000002, 'Alessandro','192.446.735-06');
Insert dbo.Cliente (codigo, nome, cpf)
values(000003, 'Bianca','135.473.291-11');
Insert dbo.Cliente (codigo, nome, cpf)
values(000004, 'Beatriz','032.665.446-03');
Insert dbo.Cliente (codigo, nome, cpf)
values(000005, 'Carlos','095.402.106-97');
Insert dbo.Cliente (codigo, nome, cpf)
values(000006, 'Otavio','273.554.109-09');
Insert dbo.Cliente (codigo, nome, cpf)
values(000007, 'Victor','402.106.443-97');
Insert dbo.Cliente (codigo, nome, cpf)
values(000008, 'Wesley','510.775.456-00');

-- TABELA FORMA DE PAGAMENTO
Insert dbo.FormaPagamento (codigo, descricao)
values (001,'A vista');
Insert dbo.FormaPagamento (codigo, descricao)
values(002,'Cartão de Credito');
Insert dbo.FormaPagamento (codigo, descricao)
values(003,'Pix');
Insert dbo.FormaPagamento (codigo, descricao)
values(004,'Cartão de Debito');
Insert dbo.FormaPagamento (codigo, descricao)
values(005,'Cheque');
Insert dbo.FormaPagamento (codigo, descricao)
values(006,'Cripto moeda');

-- TABELA SETOR
Insert dbo.Setor (codigo, nome, descricao)
values(001,'Venda','Setor responsável pelas vendas');
Insert dbo.Setor (codigo, nome, descricao)
values(002,'Limpeza','Setor responsável pela limpeza');
Insert dbo.Setor (codigo, nome, descricao)
values(003,'Gerência','Setor responsável pela admistração');
Insert dbo.Setor (codigo, nome, descricao)
values(004,'Estoque','Setor responsável pela gestão e organização dos produtos');
Insert dbo.Setor (codigo, nome, descricao)
values(005,'Contabilidade','Setor responsável pela parte financeira e economica');
Insert dbo.Setor (codigo, nome, descricao)
values(006,'Setor Geral','Setor não fixo do supermercado');
```

```
-- TABELA TELEFONE
Insert dbo.Telefone (codigo, numFone01, numFone02)
values(000001, '83999633946', '83994139614');
Insert dbo.Telefone (codigo, numFone01, numFone02)
values(000002, '83993433286', '83996839194');
Insert dbo.Telefone (codigo, numFone01, numFone02)
values(000003, '83997463656', '83994856114');
Insert dbo.Telefone (codigo, numFone01, numFone02, numFone03)
values(000004, '83993463766', '83994856244', '84996127432');
Insert dbo.Telefone (codigo, numFone01, numFone02, numFone03)
values(000005, '83993468462', '83994967344', '84998838432');
Insert dbo.Telefone (codigo, numFone01, numFone02, numFone03)
values(000006, '83994468486', '83996367194', '84998638497');
Insert dbo.Telefone (codigo, numFone01)
values(000007, '83996129614');
Insert dbo.Telefone (codigo, numFone01)
values(000008, '83993711468');

-- TABELA CEP
Insert dbo.CEP (codigo, cep)
values(000001,'75362148');
Insert dbo.CEP (codigo, cep)
values(000002,'02837462');
Insert dbo.CEP (codigo, cep)
values(000003,'53920465');
Insert dbo.CEP (codigo, cep)
values(000004,'36485921');
Insert dbo.CEP (codigo, cep)
values(000005,'37592431');

--TABELA BAIRRO
Insert dbo.Bairro (codigo, nome)
values(000001, 'Bancários');
Insert dbo.Bairro (codigo, nome)
values(000002, 'Centro');
Insert dbo.Bairro (codigo, nome)
values(000003, 'Jardim América');
Insert dbo.Bairro (codigo, nome)
values(000004, 'Pasto Novo');
Insert dbo.Bairro (codigo, nome)
values(000005, 'Santa Cruz');

-- TABELA CIDADE
Insert dbo.Cidade (codigo, nome)
values(000001, 'João Pessoa');
Insert dbo.Cidade (codigo, nome)
values(000002, 'Pedras de fogo');
Insert dbo.Cidade (codigo, nome)
values(000003, 'Cabedelo');
Insert dbo.Cidade (codigo, nome)
values(000004, 'Bayeux');
Insert dbo.Cidade (codigo, nome)
values(000005, 'Santa Rita');
```

```
--TABELA ESTADO
Insert dbo.Estado (codigo, nome)
values(000001, 'Paraíba');
Insert dbo.Estado (codigo, nome)
values(000002, 'Pernambuco');
Insert dbo.Estado (codigo, nome)
values(000003, 'Piauí');
Insert dbo.Estado (codigo, nome)
values(000004, 'Ceará');
Insert dbo.Estado (codigo, nome)
values(000005, 'Alagoas');

--TABELA PAÍS
Insert dbo.Pais (codigo, nome)
values(000001, 'Brasil');
Insert dbo.Pais (codigo, nome)
values(000002, 'Peru');
Insert dbo.Pais (codigo, nome)
values(000003, 'Argentina');
Insert dbo.Pais (codigo, nome)
values(000004, 'Estados Unidos');
Insert dbo.Pais (codigo, nome)
values(000005, 'Equador');

--TABELA ENDEREÇO
Insert dbo.Endereco (codigo, rua, numero, cep, bairro, cidade, estado, pais, complemento)
values(000001, 'tenente Mota', 72, 000001, 000001, 000001, 000001, 000001, 'Em frente ao
correio.');
Insert dbo.Endereco (codigo, rua, numero, cep, bairro, cidade, estado, pais, complemento)
values(000002, 'imbuía', 105, 000002, 000002, 000002, 000002, 000002, 'Em frente a
praça.');
Insert dbo.Endereco (codigo, rua, numero, cep, bairro, cidade, estado, pais, complemento)
values(000003, 'jacaré', 56, 000003, 000003, 000003, 000003, 000003, 'Ao lado do
mercado.');
Insert dbo.Endereco (codigo, rua, numero, cep, bairro, cidade, estado, pais, complemento)
values(000004, 'maria de Nazaré', 89, 000004, 000004, 000004, 000004, 000004, 'Casa
amarela com portão branco.');
Insert dbo.Endereco (codigo, rua, numero, cep, bairro, cidade, estado, pais, complemento)
values(000005, 'tibiri', 106, 000005, 000005, 000005, 000005, 000005, 'Em frente a
delegacia.');

--TABELA FUNCIONARIO
Insert dbo.Funcionario (codigo, nome, salario, funcao, cpf, telefone, endereco, setor)
values(000001, 'Fabiana da Conceição', 1500, 'Vendedor', '467.482.417-47', 000001, 000001,
000001);
Insert dbo.Funcionario (codigo, nome, salario, funcao, cpf, telefone, endereco, setor)
values(000002, 'Vinicius Cruz', 1350, 'Zelador', '158.448.247-34', 000002, 000002, 000002);
Insert dbo.Funcionario (codigo, nome, salario, funcao, cpf, telefone, endereco, setor)
values(000003, 'Mariana da Silva', 3000, 'Gerente', '584.738.177-14', 000003, 000003,
000003);
Insert dbo.Funcionario (codigo, nome, salario, funcao, cpf, telefone, endereco, setor)
values(000004, 'José Fagundes Silva', 1350, 'Estoquista', '554.682.147-74', 000004, 000004,
000004);
Insert dbo.Funcionario (codigo, nome, salario, funcao, cpf, telefone, endereco, setor)
values(000005, 'João Fábio Filho', 2000, 'Contador', '268.414.457-39', 000005, 000005,
000005);
```

```
--TABELA VENDA
Insert dbo.Venda (codigo, valorTotal, cliente, formaPagamento, funcionario)
Values (000001, 55.4, 000001, 001, 000001);
Insert dbo.Venda (codigo, valorTotal, cliente, formaPagamento, funcionario)
Values (000002, 23.8, 000002, 002, 000001);
Insert dbo.Venda (codigo, valorTotal, cliente, formaPagamento, funcionario)
Values (000003, 77.94, 000003, 003, 000001);
Insert dbo.Venda (codigo, valorTotal, cliente, formaPagamento, funcionario)
Values (000004, 69.96, 000004, 004, 000001);
Insert dbo.Venda (codigo, valorTotal, cliente, formaPagamento, funcionario)
Values (000005, 26.97, 000005, 001, 000001);

--TABELA ITENS VENDIDOS
Insert dbo.ItensVendidos (codigo, produto, quantProduto, venda)
values(000001, 000001, 2, 000001);
Insert dbo.ItensVendidos (codigo, produto, quantProduto, venda)
values(000002, 000002, 1, 000002);
Insert dbo.ItensVendidos (codigo, produto, quantProduto, venda)
values(000003, 000003, 6, 000003);
Insert dbo.ItensVendidos (codigo, produto, quantProduto, venda)
values(000004, 000004, 4, 000004);
Insert dbo.ItensVendidos (codigo, produto, quantProduto, venda)
values(000005, 000005, 3, 000005);
```

11. EXPLICAÇÃO DO SCRIPT DE CRIAÇÃO DO BANCO

- Ao todo temos 14 tabelas criadas que são: Produto, Cliente, FormaPagamento, Setor, Telefone, Cep, Bairro, Cidade, Estado, País, Endereco, Funcionario, Venda, ItensVendidos. Cada uma com seus respectivos atributos;

- Cada uma das 14 tabelas possui um código identificador como sua primary key;

- No total 11 tabelas possuem algum tipo de relacionamento com alguma outra tabela;

- Por fim, o banco possui 3 tabelas que possuem uma chave candidata, são elas: Cliente, Cep e Funcionario;

- As chaves candidatas dessas tabelas são respectivamente: cliente_cpf , cep_cep e funcionario_cp;

- A tabela “ItensVendidos” existe por conta da cardinalidade “1:N “ entre a tabela “Produto” e a tabela “Venda”;

12. CONSULTAS SQL NO BANCO DE DADOS

```
--1 Os clientes que mais gastaram
select a.codigo, a.nome, sum(b.valorTotal) as GastoCliente
from cliente as a join Venda as b
on a.codigo = b.cliente
group by a.codigo, a.nome order by GastoCliente desc
```

Esse código consiste em exibir os clientes que mais gastaram em ordem decrescente, é usado o **SELECT** para exibição das colunas “código” e “nome” da tabela “cliente” e agregação de soma(sum) na coluna “ValorTotal” da tabela “Venda” para exibir o todos os valores totais de venda somado. **FROM** para pegar as colunas das tabelas desejadas, **JOIN** para unir duas tabelas diferentes, **ON** para fazer a conexão das duas tabelas com a chave em comum, **GROUP BY** para a agregação de soma ser agrupada pelo código e nome do cliente e **ORDER BY** para ordenar a coluna de agregação de forma decrescente com **DESC.**

Resultado:

	codigo	nome	GastoCliente
1	3	Bianca	77,94
2	4	Beatriz	69,96
3	1	Alexandre	55,40
4	5	Carlos	26,97
5	2	Alessandro	23,80

Fonte: Autores, SQL Server

```
--2 Os produtos mais vendidos
select a.nome, a.categoria, a.precoVenda, b.quantProduto
from Produto as a join ItensVendidos as b
on a.codigo = b.produto
order by b.quantProduto desc
```

Esse código consiste em exibir os produtos mais vendidos em ordem decrescente, usado o **SELECT** para exibição das colunas “nome”, “categoria”, “precoVenda” da tabela “produto” e “quantProduto” da tabela “ItensVendidos. O **FROM** para pegar as colunas das tabelas desejadas, **JOIN** para unir duas tabelas diferentes, **ON** para fazer a conexão das duas tabelas com a chave em comum e **ORDER BY** para ordenar as colunas de forma decrescente com **DESC.**

Resultado:

	nome	nome	categoria	precoVenda	quantProduto
1	Hambúrguer Seara	Hambúrguer Seara	Alimento	12,99	6
2	Leite Guarabira	Leite Guarabira	Alimento	17,49	4
3	Guaraná Jesus	Guaraná Jesus	Alimento	8,99	3
4	Arroz Mundial	Arroz Mundial	Alimento	27,70	2
5	Feijão Mundial	Feijão Mundial	Alimento	23,80	1

Fonte: Autores, SQL Server

```sql
--3 mostra os funcionários com seus respectivos setores
select a.codigo, a.nome, a.funcao, b.nome
from Funcionario as a left join Setor as b
on a.setor = b.codigo
```

Esse código consiste em exibir cada funcionário em seu setor contratado, usado o **SELECT** para exibição das colunas “código”, “nome” e “função” da tabela “Funcionários” e “nome” da tabela “Setor”. O **FROM** para pegar as colunas das tabelas utilizadas, **LEFT JOIN** para unir duas tabelas diferentes com todos os resultados da tabela esquerda independente se a dados na tabela da direita, **ON** para fazer a conexão das duas tabelas com a chave em comum.

Resultado:

	codigo	nome	funcao	nome
1	1	Fabiana da Conceição	Vendedor	Venda
2	2	Vinicius Cruz	Zelador	Limpeza
3	3	Mariana da Silva	Gerente	Gerência
4	4	José Fagundes Silva	Estoquista	Estoque
5	5	João Fábio Filho	Contador	Contabilidade

Fonte: Autores, SQL Server

```sql
--4 mostra o(s) setor(es) sem funcionários
select a.nome, b.nome
from Setor as a left join Funcionario as b
on b.setor = a.codigo
where b.nome is null
```

Esse código consiste em exibir o(s) setor(es) que ainda não possui algum funcionário ocupando algum tipo de cargo, usado o **SELECT** para exibição das colunas “nome” da tabela “Setor” e “nome” da tabela “Funcionários”. O **FROM** para pegar as colunas das tabelas utilizadas, **LEFT JOIN** para unir duas tabelas diferentes com todos os resultados da tabela esquerda independente se a dados na tabela da direita, **ON** para fazer a conexão das duas tabelas com a chave em comum, **WHERE** para darmos uma condição de apenas resultados nulos na coluna “nome” da tabela “Funcionários”, assim trazendo apenas os setores que está sem funcionários.

Resultado:

	nome	nome
1	Setor Geral	NULL

Fonte: Autores, SQL Server

```sql
--5 mostra os clientes que gastaram entre 20.00 e 60.00 reais em compras
select a.codigo, a.nome, b.valorTotal as GastoCliente
from cliente as a join Venda as b
on a.codigo = b.cliente
where valorTotal between 25.00 and 60.00
order by GastoCliente desc
```

Esse código consiste em exibir clientes que gastaram entre R$ 20,00 e R$ 60,00, **SELECT** para exibição das colunas “código”, “nome” da tabela “cliente” e “ValorTotal” da tabela “Venda”. O **FROM** para pegar as colunas das tabelas desejadas, **JOIN** para unir duas tabelas diferentes, **ON** para fazer a conexão das duas tabelas com a chave em comum, **WHERE** para darmos uma condição de valores exigidos com auxilio do **BETWEEN** para colocarmos ENTRE os valores requisitados.

Resultado:

	codigo	nome	GastoCliente
1	1	Alexandre	55,40
2	5	Carlos	26,97

Fonte: Autores, SQL Server

```
--6 produtos menos vendidos (produtos vendidos menos de 3 vezes)
select a.nome, a.precoVenda, b.quantProduto
from Produto as a join ItensVendidos as b
on a.codigo = b.produto where b.quantProduto < 3
order by b.quantProduto desc
```

Esse código consiste em exibir os produtos vendidos menos de 3 vezes, **SELECT** para exibição das colunas “nome”, “precoVenda da tabela “Produtos” e “quantProduto” da tabela “ItensVendidos”. O **FROM** para pegar as colunas das tabelas desejadas, **JOIN** para unir duas tabelas diferentes, **ON** para fazer a conexão das duas tabelas com a chave em comum, **WHERE** para aplicar a condição da quantidade de produtos vendidos menor que 3 e ordena-las.

Resultado:

	nome	precoVenda	quantProduto
1	Arroz Mundial	27,70	2
2	Feijão Mundial	23,80	1

Fonte: Autores, SQL Server

```
--7 Estado de cada funcionário
select a.codigo, a.nome, c.nome
from Funcionario as a join Endereco as b
on a.endereco = b.codigo join Estado as c
on b.estado = c.codigo
```

Esse código consiste em exibir o estado de cada funcionário, **SELECT** para exibição das colunas “código”, “nome” da tabela “Funcionário” e “nome” da tabela “Estado”. O **FROM** para pegar as colunas das tabelas desejadas, **JOIN** para unir duas tabelas diferentes, **ON** para fazer a conexão das duas tabelas com a chave em comum e novamente usa o **JOIN** para unir a terceira tabela juntamente com o **ON** para fazer a conexão com a terceira tabela.

Resultado:

	codigo	nome	nome
1	1	Fabiana da Conceição	Paraíba
2	2	Vinicius Cruz	Pernambuco
3	3	Mariana da Silva	Piauí
4	4	José Fagundes Silva	Ceará
5	5	João Fábio Filho	Alagoas

Fonte: Autores, SQL Server

```
--8 valor de vendas de acordo com cada forma de pagamento
select a.descricao, sum(b.valorTotal) as valorDeVenda
from FormaPagamento as a join Venda as b
on a.codigo = b.formaPagamento
group by a.descricao
```

Esse código consiste em exibir o valor total apurado nas vendas de acordo com cada forma de pagamento, **SELECT** para exibição das colunas “descrição” da tabela “FormaPagamento” agregação de **SUM** para “valorTotal” da tabela “Venda”. O **FROM** para pegar as colunas das tabelas desejadas, **JOIN** para unir duas tabelas diferentes, **ON** para fazer a conexão das duas tabelas com a chave em comum e sempre que tiver agregação usar o **GROUP BY** para agrupar a tabela pela “descrição”.

Resultado:

	descricao	valorDeVenda
1	A vista	82,37
2	Cartão de Credito	23,80
3	Cartão de Debito	69,96
4	Pix	77,94

Fonte: Autores, SQL Server

```
--9 saber qual forma de pagamento recebeu o maior valor de venda
select top 1 a.descricao, sum(b.valorTotal) as valorDeVenda
from FormaPagamento as a join Venda as b
on a.codigo = b.formaPagamento
group by a.descricao
order by valorDeVenda desc
```

Esse código consiste em exibir qual forma de pagamento recebeu maio quantidade de valor em venda, **SELECT com TOP (limitando a exibição em apenas um registro),** para exibição das colunas “descrição” da tabela “FormaPagamento” agregação de **SUM** para “valorTotal” da tabela “Venda”. O **FROM** para pegar as colunas das tabelas desejadas, **JOIN** para unir duas tabelas diferentes, **ON** para fazer a conexão das duas tabelas com a chave em comum e sempre que tiver agregação usar o **GROUP BY** para agrupar a tabela pela “descrição” com **ORDER BY** para ordenar de forma decrescente com **DESC**.

Resultado:

	descricao	valorDeVenda
1	A vista	82,37

Fonte: Autores, SQL Server

```sql
--10 produtos que não foram vendidos
select a.nome, b.venda as ProdutoSemVenda
from Produto as a left join ItensVendidos as b
on a.codigo = b.produto
where b.venda is null
```

Esse código consiste em exibir os produtos que não foram vendidos, **SELECT** para exibição das colunas “nome” da tabela “Produto” e “venda” da tabela “ItensVendidos”. O **FROM** para pegar as colunas das tabelas utilizadas, **LEFT JOIN** para unir duas tabelas diferentes com todos os resultados da tabela esquerda independente se a dados na tabela da direita, **ON** para fazer a conexão das duas tabelas com a chave em comum, **WHERE** com a condição **IS NULL** da coluna venda quando for nula, assim trará os produtos que ainda não foram registrados na tabela de venda.

Resultado:

	nome	ProdutoSemVenda
1	Chocolate Garoto	NULL
2	Macarrão MassaDaBoa	NULL
3	Detergente LimpaGeral	NULL
4	Sabonete Senador	NULL
5	Sabão em pó Omo	NULL

Fonte: Autores, SQL Server

13. CONSIDERAÇÕES FINAIS

Neste trabalho obtemos uma base essencial para o desenvolvimento de Bancos de Dados, desde modelos conceituais, onde é visto todo o conceito do banco de dados muito antes de sua criação, modelo lógico, onde é visto a lógica do banco e suas propriedades, mais próximo de um banco de dados e a criação do próprio banco com suas consultas.

Com o Big Data é notável a importância dos dados atualmente, levando as grandes empresas a decisões de negócios antecipadamente através da ciência de dados, tendo informações necessárias para o crescimento ou até mesmo evitar graves erros futuros.

14. REFERÊNCIAS BIBLIOGRÁFICAS

Heuser, Carlos Alberto. Projeto de banco de dados / Carlos Alberto Heuser. – 6. ed. – Dados eletrônicos. – Porto Alegre : Bookman, 2009.