ESCOLA E FACULDADE DE TECNOLOGIA SENAI GASPAR RICARDO JÚNIOR

DESENVOLVIMENTO DE SISTEMAS

FELIPE MARQUES

GABRIEL RIBEIRO DE CAMARGO MIRANDA

JOÃO VICTOR DE SOUZA SANTOS

LUCAS MIGUEL LEITE

PROFESSOR ANDRÉ CASSULINO ARAÚJO SOUZA

BANCO DE DADOS

BANCO E-COMMERCE

SOROCABA

20/06/2025

LISTA DE ILUSTRAÇÕES

FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Este trabalho tem como tema o desenvolvimento de um sistema de gerenciamento de vendas (e-commerce), contemplando funcionalidades fundamentais como carrinho de compras, realização de pedidos, cadastro de produtos e categorias. Embora existam diversas outras funcionalidades possíveis, como busca, recomendação e rastreamento de produtos, optamos por focar nos elementos essenciais para o funcionamento básico de uma plataforma de comércio eletrônico.

O principal objetivo deste projeto é aplicar os conhecimentos adquiridos nas aulas de Banco de Dados, ministradas pelo professor André, por meio da criação da estrutura de um sistema de e-commerce utilizando um banco de dados relacional MySQL.

O banco de dados contempla as entidades necessárias para o funcionamento do sistema. A entidade `usuario` armazena informações pessoais e de autenticação. Cada usuário pode possuir um carrinho, composto por diversos `item_carrinho`, os quais se referem a produtos disponíveis. Os produtos são organizados por categorias e podem possuir múltiplas `foto_produto`. Além disso, usuários podem realizar pedidos, compostos por `item_pedido`, registrando o histórico de compras realizadas.

2 MODELAGEM CONCEITUAL

Este capítulo descreve detalhadamente a estrutura do banco de dados desenvolvida para o sistema de gerenciamento de vendas (e-commerce), com base no modelo Entidade-Relacionamento (DER). São apresentados os principais elementos do modelo: entidades, atributos e relacionamentos, bem como o diagrama que representa visualmente essas estruturas.

2.1 Descrição das Entidades, Atributos e Relacionamentos

O banco de dados foi modelado para contemplar as principais operações de um sistema de e-commerce, incluindo cadastro de usuários, gerenciamento de produtos e categorias, carrinho de compras, pedidos e imagens dos produtos. A seguir, descrevem-se as entidades envolvidas:

2.1.1 Entidade `usuario`

Responsável por armazenar os dados dos usuários cadastrados na plataforma.

- Atributos:
  - `id`, `primeiroNome`, `ultimoNome`, `email`, `senha`;
  - `ativo`, `funcionario`, `superUsuario`, `verificado`;
  - `ultimoLogin`, `criadoEm`, `atualizadoEm`.
- Relacionamentos:
  - Um `usuario` possui um `carrinho` (1:1);
  - Um `usuario` pode realizar vários `pedidos` (1:N).

2.1.2 Entidade `carrinho`

Representa o carrinho de compras associado a um usuário.

- Atributos:
  - `id`, `fkusuarioId`, `criadoEm`, `atualizadoEm`.
- Relacionamentos:

- Um `carrinho` pertence a um `usuario` (1:1);
- Um `carrinho` contém vários `item_carrinho` (1:N).

2.1.3 Entidade `item_carrinho`

Itens adicionados ao carrinho por um usuário.

- Atributos:
  - `id`, `fkcarrinhoId`, `fkprodutoId`, `quantidade`, `preco`;
  - `criadoEm`, `atualizadoEm`.
- Relacionamentos:
  - Um `item_carrinho` pertence a um `carrinho` (N:1);
  - Um `item_carrinho` faz referência a um `produto` (N:1).

2.1.4 Entidade `pedido`

Representa uma compra finalizada por um usuário.

- Atributos:
  - `id`, `fkusuarioId`, `status`, `valorTotal`;
  - `criadoEm`, `atualizadoEm`.
- Relacionamentos:
  - Um `pedido` pertence a um `usuario` (N:1);
  - Um `pedido` contém vários `item_pedido` (1:N).

2.1.5 Entidade `item_pedido`

Itens que compõem um pedido realizado.

- Atributos:
  - `id`, `fkpedidoId`, `fkprodutoId`, `quantidade`, `preco`;
  - `criadoEm`, `atualizadoEm`.
- Relacionamentos:
  - Um `item_pedido` pertence a um `pedido` (N:1);
  - Um `item_pedido` faz referência a um `produto` (N:1).

2.1.6 Entidade `categoria`

Classifica os produtos em grupos.

- Atributos:
  - `id`, `nome`, `criadoEm`, `atualizadoEm`.
- Relacionamentos:
  - Uma `categoria` possui vários `produtos` (1:N).

2.1.7 Entidade `produto`

Contém informações sobre os produtos disponíveis na plataforma.

- Atributos:
  - `id`, `nome`, `descricao`, `nomeUrl`, `subCategoria`;
  - `fkcategoriaId`, `estoque`, `preco`, `promocao`;
  - `criadoEm`, `atualizadoEm`.
- Relacionamentos:
  - Um `produto` pertence a uma `categoria` (N:1).;
  - Um `produto` pode estar presente em vários `item_carrinho` (1:N).;
  - Um `produto` pode estar presente em vários `item_pedido` (1:N).;
  - Um `produto` possui várias `foto_produto` (1:N).

2.1.8 Entidade `foto_produto`

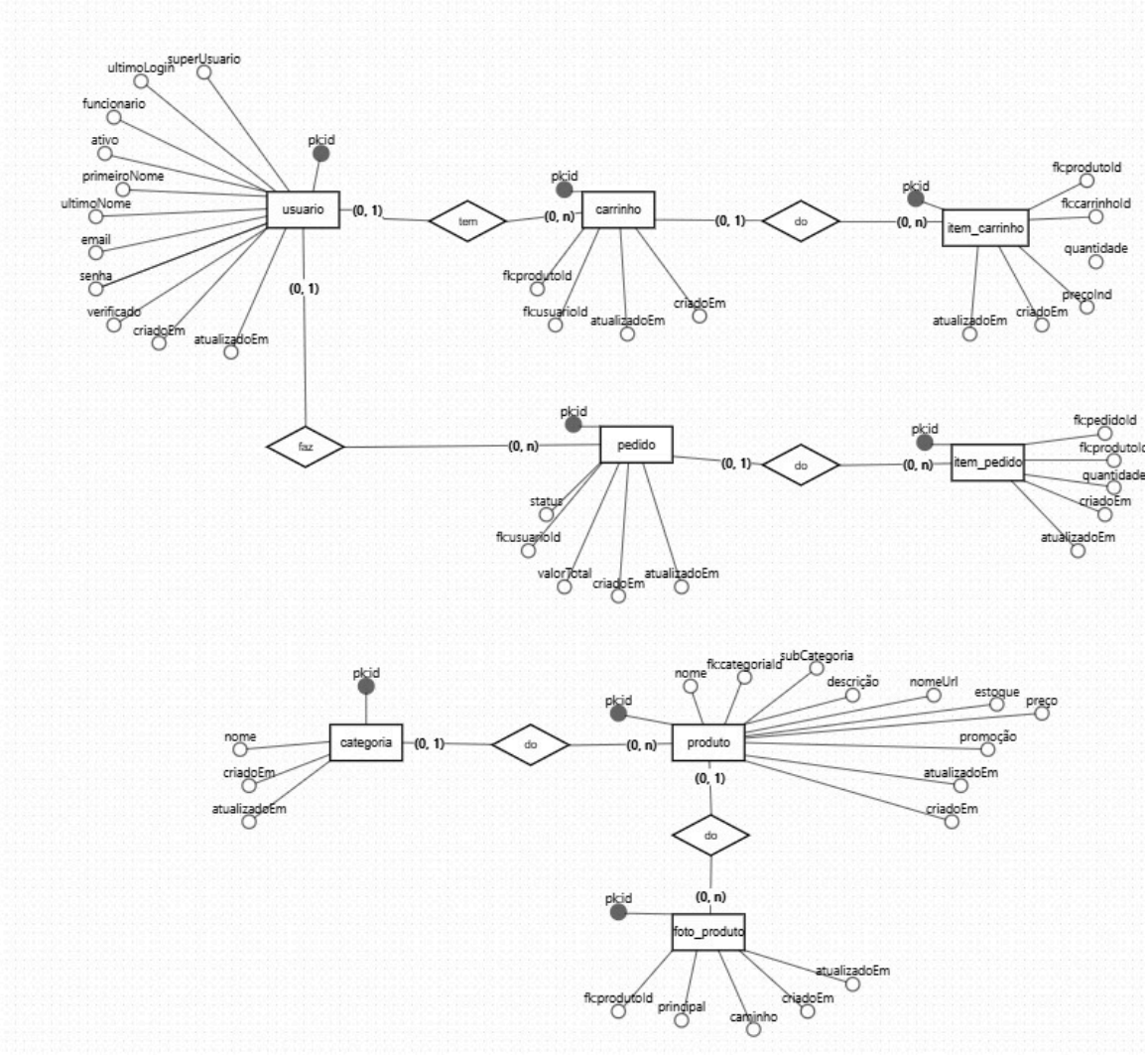
Imagens associadas aos produtos.

- Atributos:
  - `id`, `fkprodutoId`, `caminho`, `principal`;
  - `criadoEm`, `atualizadoEm`.
- Relacionamentos:
  - Uma `foto_produto` pertence a um `produto` (N:1).

2.2 Diagrama Entidade-Relacionamento (DER)

A Figura 1 apresenta o Diagrama Entidade-Relacionamento (DER) que representa graficamente a estrutura do banco de dados descrita acima. O diagrama demonstra as entidades, seus respectivos atributos e os relacionamentos entre elas, com as cardinalidades definidas entre os pares de entidades.

Figura 1 – Diagrama Entidade-Relacionamento do sistema de e-commerce



Fonte: Elaborado pelo grupo (2025).

3 MODELAGEM LÓGICA

A modelagem lógica é a etapa em que o modelo Entidade-Relacionamento (DER) é transformado em um modelo relacional, com o objetivo de estruturar os dados em tabelas normalizadas adequadas para implementação em um Sistema Gerenciador de Banco de Dados (SGBD). Neste projeto, optou-se pela utilização do MySQL como SGBD, o que influenciou algumas decisões de estrutura.

3.1 Transformação do DER em Modelo Relacional

Com base no DER apresentado anteriormente, foram criadas tabelas correspondentes a cada entidade, bem como chaves primárias e estrangeiras para garantir a integridade referencial entre os dados. A seguir, descrevem-se as principais transformações:

3.1.1 Tabelas e Chaves

- A entidade `usuario` foi transformada na tabela `Usuario`, com `id` como chave primária. As permissões (superUsuario, funcionario) foram mantidas como atributos booleanos, facilitando o controle de acesso;
- A entidade `carrinho` deu origem à tabela `Carrinho`, contendo uma chave estrangeira `usuarioId` referenciando `Usuario(id)`, estabelecendo um relacionamento 1:1;
- A entidade `item_carrinho` tornou-se a tabela `ItemCarrinho`, com chaves estrangeiras `carrinhoId` (para carrinho) e `produtoId` (para produto), formando um relacionamento N:1 com ambas;
- A entidade `produto` originou a tabela `Produto`, com `id` como chave primária. A associação com categoria é feita por meio da chave estrangeira `categoriaId` referenciando `Categoria(id)`;
- A entidade `foto_produto` virou a tabela `FotoProduto`, com a chave estrangeira `produtoId` referenciando `Produto(id)`;

- As entidades `pedido` e `item_pedido` formam uma estrutura similar à do `carrinho`. Sendo que `Pedido` contém `usuarioId` e `ItemPedido` contém `pedidoId` e `produtoId`, referenciando respectivamente `Usuario(id)`, `Pedido(id)` e `Produto(id)`, definindo seus relacionamentos;
- A tabela `Categoria` é referenciada por produto via chave estrangeira, permitindo a classificação de produtos.

3.2 Normalização

O processo de normalização foi aplicado com o objetivo de eliminar redundâncias e garantir a integridade dos dados. Todas as tabelas foram estruturadas, no mínimo, até a Terceira Forma Normal (3FN), conforme descrito a seguir:

3.2.1 Primeira Forma Normal (1FN)

Todas as tabelas possuem atributos atômicos, ou seja, que não podem ser decompostos. Não há atributos multivalorados nem repetição de grupos de atributos.

3.2.2 Segunda Forma Normal (2FN)

Foi evitada a presença de dependências parciais em tabelas com chaves compostas. Como todas as tabelas possuem chaves primárias simples (com exceção de tabelas intermediárias como `item_carrinho` e `item_pedido`), as dependências foram corretamente associadas à chave inteira.

3.2.3 Terceira Forma Normal (3FN)

Todas as dependências transitivas foram eliminadas. Por exemplo:

- Informações de categorias estão separadas da tabela de produtos;
- Detalhes dos usuários estão separados dos carrinhos e pedidos;
- Imagens dos produtos estão em uma tabela específica (foto_produto), ligadas por chave estrangeira.

3.2.4 Justificativa para a Escolha da 3FN

A adoção da Terceira Forma Normal (3FN) se deu por ser suficiente para a maioria dos sistemas comerciais, garantindo:

- Clareza e organização do esquema relacional;
- Evitação de anomalias de inserção, atualização e exclusão;
- Facilidade de manutenção e expansão do sistema.

Além disso, a estrutura 3FN permite posterior otimização por desnormalização, caso haja necessidade de ganho de performance em consultas específicas (prática comum em sistemas de e-commerce de maior escala).

4 ESTRUTURA DO BANCO DE DADOS

Neste capítulo é apresentada a estrutura do banco de dados implementado para o sistema de e-commerce, incluindo os scripts de definição das tabelas (DDL), a descrição dos campos de cada tabela e os relacionamentos estabelecidos entre elas.

4.1 Scripts de DDL

A estrutura foi criada por meio de comandos SQL da linguagem de definição de dados (DDL), utilizando o SGBD MySQL. O script contempla a criação das tabelas, definição de chaves primárias, estrangeiras e permissões de usuários. Os códigos SQL encontram-se integralmente no **Apêndice A**.

4.2 Descrição das Tabelas

A seguir, apresentam-se as principais tabelas criadas no banco de dados, com seus respectivos campos e finalidades.

4.2.1 Tabela `Usuario`

Armazena os dados dos usuários do sistema, incluindo clientes e administradores.

- id: identificador único do usuário (chave primária);
- primeiroNome: primeiro nome do usuário;
- ultimoNome: sobrenome do usuário;
- email: endereço de e-mail;
- senha: senha criptografada;
- verificado: indica se o e-mail foi verificado (0 ou 1);
- ativo: indica se a conta está ativa (0 ou 1);
- funcionario: identifica se o usuário é um funcionário (0 ou 1);
- superUsuario: define se o usuário tem privilégios administrativos;
- ultimoLogin: data e hora do último acesso;

- criadoEm: data de criação do registro;
- atualizadoEm: data da última atualização do registro.

4.2.2 Tabela `Produto`

Contém informações sobre os produtos disponíveis para compra.

- id: identificador único do produto (chave primária);
- nome: nome do produto;
- categoriaId: chave estrangeira que relaciona o produto à sua categoria;
- subCategoria: subcategoria do produto;
- descricao: descrição detalhada do produto;
- nomeUrl: nome amigável para uso em URLs;
- estoque: quantidade disponível em estoque;
- preco: preço atual do produto;
- promocao: indica se o produto está em promoção (0 ou 1);
- criadoEm: data de criação do registro;
- atualizadoEm: data da última atualização do registro.

4.2.3 Tabela `Categoria`

Utilizada para classificar os produtos em grupos.

- id: identificador único da categoria (chave primária);
- nome: nome da categoria;
- criadoEm: data de criação do registro;
- atualizadoEm: data da última atualização do registro.

4.2.4 Tabela `Carrinho`

Representa o carrinho de compras de cada usuário.

- id: identificador do carrinho (chave primária);
- usuarioId: chave estrangeira para o usuário proprietário do carrinho;
- produtoId: chave estrangeira para o produto;
- criadoEm: data de criação do registro;

- atualizadoEm: data da última atualização do registro.

4.2.5 Tabela `ItemCarrinho`

Armazena os produtos adicionados ao carrinho.

- id: identificador único do item (chave primária);
- carrinhoId: chave estrangeira para o carrinho;
- produtoId: chave estrangeira para o produto;
- quantidade: quantidade do produto no carrinho;
- precoInd: preço unitário do produto no momento da adição;
- criadoEm: data de criação do registro;
- atualizadoEm: data da última atualização do registro.

4.2.6 Tabela `Pedido`

Registra os pedidos realizados pelos usuários.

- id: identificador único do pedido (chave primária);
- status: status atual do pedido (ex: “pendente”, “enviado”);
- usuarioId: chave estrangeira para o usuário que realizou o pedido;
- valorTotal: valor total da compra;
- criadoEm: data de criação do registro;
- atualizadoEm: data da última atualização do registro.

4.2.7 Tabela `ItemPedido`

Contém os produtos associados a cada pedido.

- id: identificador único do item (chave primária);
- pedidoId: chave estrangeira para o pedido;
- produtoId: chave estrangeira para o produto adquirido;
- quantidade: quantidade comprada do produto;
- criadoEm: data de criação do registro;
- atualizadoEm: data da última atualização do registro.

4.2.8 Tabela `FotoProduto`

Armazena as imagens relacionadas a cada produto.

- id: identificador único da foto (chave primária);
- produtoId: chave estrangeira que associa a imagem ao produto;
- principal: indica se é a imagem principal do produto (0 ou 1);
- caminho: endereço da imagem no sistema;
- criadoEm: data de criação do registro;
- atualizadoEm: data da última atualização do registro.

4.3 Relacionamento entre Tabelas

Os relacionamentos entre as tabelas foram implementados por meio de chaves estrangeiras, garantindo a integridade referencial. Abaixo estão os principais relacionamentos definidos:

- Usuario 1:N Pedido;
- Usuario 1:N Carrinho;
- Carrinho 1:N ItemCarrinho;
- Produto 1:N ItemCarrinho, ItemPedido, FotoProduto;
- Pedido 1:N ItemPedido;
- Categoria 1:N Produto.

Essas conexões refletem o modelo lógico previamente discutido e foram implementadas com ALTER TABLE e FOREIGN KEY.

5 MANIPULAÇÃO DE DADOS

Este capítulo descreve os principais comandos utilizados para a manipulação de dados no banco de dados desenvolvido, com foco nas operações de inserção, atualização, deleção e consulta.

5.1 Inserção de Dados

As inserções foram realizadas por meio de procedimentos armazenados, garantindo que múltiplas operações fossem executadas de forma transacional. O procedimento `realizar_insercoes`, por exemplo, insere dados nas tabelas `Categoria`, `Usuário`, `Produto`, `Pedido`, `ItemPedido`, `Carrinho` e `ItemCarrinho`, compondo um fluxo completo de registro e compra de produtos. A utilização de transações (comandos DML) assegura a persistência e integridade dos dados, permitindo o cancelamento automático de todas as operações em caso de erro, por meio da instrução ROLLBACK. Este aspecto da manipulação de dados será explorado em maior profundidade no Capítulo 7. O procedimento completo encontra-se disponível no **Apêndice B**.

5.2 Atualização de Dados

As atualizações de dados também foram realizadas com auxílio de `procedures`, utilizando comandos UPDATE dentro de transações. O procedimento `realizar_atualizacoes` realiza, por exemplo:

- A modificação do nome da categoria de ID 1;
- A redução do estoque de um produto após a compra;
- A alteração do status de um pedido de “Pendente” para “Enviado”;
- A atualização do campo `ultimoLogin` de um usuário.

Essas alterações refletem ações típicas de um sistema de e-commerce e demonstram como o banco lida com o dinamismo das operações comerciais. O código completo pode ser consultado no **Apêndice C**.

5.3 Deleção de Dados

A exclusão de dados é igualmente sensível, especialmente em sistemas com múltiplas tabelas relacionadas. Para garantir a integridade referencial, a ordem de deleção foi cuidadosamente planejada no procedimento `realizar_delecoes`.

Esse procedimento remove registros de carrinhos, itens de pedido, pedidos, produtos, usuários e categorias, respeitando a sequência correta para evitar conflitos com chaves estrangeiras. O uso de transações assegura que qualquer falha no processo reverta todas as exclusões realizadas até então. O script completo está disponível no **Apêndice D**.

5.4 Consultas DQL

As consultas DQL (Data Query Language), representadas principalmente pelo comando SELECT, são utilizadas para interrogar e recuperar dados armazenados nas tabelas do banco. A seguir, são apresentados exemplos práticos de consultas SQL aplicadas ao banco de dados do sistema, as quais visam extrair informações úteis para análise de desempenho, comportamento dos usuários e acompanhamento das vendas. Os scripts completos podem ser consultados no **Apêndice E**.

As consultas foram elaboradas para responder a questões comuns em um sistema de e-commerce. Abaixo, descrevem-se algumas delas:

- Consulta 1 – Quantidade de pedidos por status: Esta consulta retorna a quantidade total de pedidos agrupados por seu respectivo status (por exemplo: pendente, enviado, cancelado, etc.). Trata-se de uma métrica importante para o acompanhamento do fluxo logístico.
- Consulta 2 – Total vendido por produto: Ao agrupar os itens vendidos por nome de produto, esta consulta retorna a quantidade total vendida de cada produto, permitindo identificar os itens com maior saída. Os resultados são ordenados de forma decrescente, listando os dez produtos mais vendidos.

- Consulta 3 – Total gasto por usuário: Essa consulta calcula o número de pedidos e o valor total gasto por cada usuário. É útil para fins de análise de comportamento de compra e definição de estratégias de fidelização.
- Consulta 4 – Produtos em promoção com estoque baixo: Filtra os produtos que estão em promoção (promotion = 1) e que possuem estoque inferior a 10 unidades, o que pode indicar a necessidade de reposição ou destaque em campanhas de marketing.
- Consulta 5 – Total de produtos por categoria: Apresenta a quantidade de produtos cadastrados em cada categoria do sistema, permitindo avaliar a distribuição de itens no catálogo.
- Consulta 6 – Total de itens no carrinho por usuário: Informa o total de itens adicionados ao carrinho por cada usuário, servindo como indicador de intenção de compra ou possíveis desistências (abandono de carrinho).

Essas consultas demonstram a capacidade do banco de dados de fornecer informações estratégicas para a gestão do sistema, apoiando desde decisões operacionais até ações de marketing. A utilização adequada dos comandos DQL possibilita análises em tempo real, contribuindo para a tomada de decisões baseadas em dados — prática comum em áreas como a ciência de dados —, onde a extração e interpretação de informações estruturadas é essencial para gerar valor às organizações.

6 CONTROLE DE ACESSO (DCL)

O controle de acesso em um banco de dados relacional é uma medida fundamental para garantir a segurança da informação e o uso adequado dos recursos do sistema. Por meio dos comandos da DCL (Data Control Language), é possível conceder (GRANT) ou revogar (REVOKE) permissões específicas a diferentes usuários, conforme seus níveis de responsabilidade e necessidade de acesso.

No banco de dados desenvolvido para este sistema, foram criados diferentes perfis de usuários, com privilégios distintos conforme a função de cada um. Essa estrutura é essencial para evitar acessos indevidos, proteger dados sensíveis e preservar a integridade das operações.

6.1 Criação de Usuários

Quatro usuários foram criados com diferentes perfis:

- administrador: possui acesso total a todas as funcionalidades do banco de dados;
- cliente_01: possui permissão de leitura geral, mas está impedido de realizar operações de inserção, atualização e deleção;
- cliente_02: tem acesso limitado, com permissão de leitura e atualização apenas em determinadas tabelas;
- cliente_03: possui quase todos os privilégios, exceto o de deleção de registros.

Esses usuários foram criados com credenciais específicas, incluindo senhas definidas individualmente, o que contribui para o controle de autenticação e restringe o acesso exclusivamente a conexões originadas do próprio servidor (localhost), aumentando a segurança do sistema.

6.2 Concessão e Revogação de Privilégios

A distribuição de privilégios foi feita com o comando `GRANT`, enquanto o comando `REVOKE` foi utilizado para restringir ações indesejadas. O usuário `administrador` recebeu todos os privilégios sobre o banco de dados, permitindo a administração completa do sistema.

O usuário `cliente_01` inicialmente recebeu todos os privilégios, mas, em seguida, teve os comandos `INSERT`, `UPDATE` e `DELETE` revogados, permitindo apenas operações de consulta.

O usuário `cliente_02` recebeu permissão de leitura geral (SELECT) e permissões específicas de atualização (UPDATE) em tabelas como `Produto`, `FotoProduto`, `Carrinho`, `Pedido` e `ItemPedido`.

Já o usuário `cliente_03` teve todos os privilégios concedidos, exceto o privilégio de deleção, garantindo que seus acessos não comprometam a integridade dos dados ao excluir registros.

6.3 Importância do Controle de Acesso

A implementação de controle de acesso com comandos DCL é essencial para a governança de dados. Além de evitar ações maliciosas ou acidentais, essa estratégia contribui para a rastreabilidade de operações, definição de responsabilidades e conformidade com princípios de segurança da informação, como o princípio do menor privilégio — que garante que cada usuário tenha acesso apenas ao que é estritamente necessário para a sua função.

O script completo de criação de usuários e definição de permissões encontra-se no **Apêndice F**.

7 CONTROLE DE TRANSAÇÕES (DTL)

A Linguagem de Transação de Dados (DTL) é um conjunto de comandos utilizados para gerenciar as operações de manipulação de dados dentro de um banco de dados, garantindo que múltiplas instruções sejam executadas de forma consistente e segura, especialmente em ambientes onde concorrência e integridade são essenciais.

No contexto deste trabalho, a DTL foi aplicada através da implementação de procedimentos armazenados que englobam comandos de inserção, atualização e deleção de dados (DML), além da criação das tabelas, encapsulados dentro de transações explícitas. Isso assegura que todas as operações relacionadas a uma tarefa específica sejam concluídas integralmente ou, em caso de falhas, revertidas completamente, evitando estados inconsistentes no banco de dados.

7.1 Conceitos Fundamentais da DTL

- Transação: Conjunto indivisível de operações que devem ser realizadas como uma única unidade lógica;
- Atomicidade: Garantia de que todas as operações dentro da transação sejam concluídas com sucesso ou nenhuma delas seja aplicada;
- Consistência: O banco de dados permanece em um estado válido antes e depois da transação;
- Isolamento: As transações ocorrem de forma isolada, sem interferir umas nas outras, mantendo a integridade dos dados;
- Durabilidade: Uma vez confirmada, a transação é permanente mesmo em caso de falhas do sistema.

7.2 Uso de Transações no Trabalho

Os procedimentos armazenados criados para as operações de manipulação de dados (`realizar_insercoes`, `realizar_atualizacoes` e `realizar_delecoes`) e na criação

das tabelas, implementam explicitamente o controle transacional por meio dos comandos:

- START TRANSACTION; para iniciar a transação;
- Execução dos comandos DML (INSERT, UPDATE, DELETE), ou a criação das tabelas;
- COMMIT; para confirmar e persistir as alterações;
- ROLLBACK; para desfazer as alterações em caso de erro, acionado por handlers de exceção (DECLARE EXIT HANDLER FOR SQLEXCEPTION).

7.3 Benefícios da Utilização da DTL

A adoção da DTL proporciona:

- Integridade dos dados: evita a ocorrência de dados parcialmente inseridos ou atualizados;
- Segurança contra falhas: permite reverter alterações em caso de erros, garantindo a confiabilidade;
- Facilidade de manutenção: procedimentos armazenados centralizam a lógica transacional, facilitando atualizações e auditoria;
- Melhor controle de concorrência: ao isolar as operações dentro das transações, reduz conflitos em ambientes multiusuário.

Os scripts completos dos procedimentos que implementam essa lógica transacional estão disponíveis no **Apêndice A**, **Apêndice B, Apêndice C** e **Apêndice D**.

8 CONCLUSÃO

O presente trabalho abordou o desenvolvimento de um sistema de banco de dados para gerenciamento de e-commerce, contemplando desde a definição do esquema até a manipulação e controle de dados. Foi realizada a implementação de procedimentos armazenados para operações transacionais, garantindo a integridade e segurança das informações. Além disso, o controle de acesso foi estruturado para assegurar níveis diferenciados de permissão aos usuários, promovendo a governança dos dados.

Durante o processo de desenvolvimento, foi possível compreender a importância da organização dos dados e da aplicação correta das linguagens SQL (DDL, DML, DCL e DTL) para o funcionamento eficiente e seguro do sistema. A experiência evidenciou a necessidade de atenção especial à gestão de transações para evitar inconsistências, bem como a relevância do controle de privilégios para a proteção dos dados.

Entre as lições aprendidas, destaca-se a importância da automação por meio de procedimentos armazenados para padronizar e facilitar a manutenção das operações no banco de dados. Também foi notória a relevância da utilização de transações para garantir atomicidade e consistência durante as manipulações de dados.

Como possíveis melhorias futuras, sugere-se a implementação de mecanismos adicionais de segurança, como criptografia avançada e auditoria de acessos, além da adoção de políticas de backup e recuperação de dados mais robustas. Outra oportunidade reside na otimização das consultas e índices para aprimorar o desempenho em grandes volumes de dados.

Portanto, este projeto contribuiu significativamente para o aprofundamento do conhecimento prático em bancos de dados relacionais, além de fortalecer a compreensão das melhores práticas para o desenvolvimento de sistemas confiáveis, escaláveis e eficientes.

REFERÊNCIAS

Normas da ABNT: regras de formatação para trabalhos acadêmicos. Disponível em: <https://www.todamateria.com.br/normas-abnt-trabalhos/>. Acesso em: 19 jun. 2025.

SOUZA, A. **Banco de dados at main · profAndreSouza/Material**. Disponível em: <https://github.com/profAndreSouza/Material/tree/main/Banco%20de%20Dados>. Acesso em: 19 jun. 2025.

Projeto E-Commerce - Banco de Dados MySQL. Disponível em: <https://github.com/Trabalho-BD/banco_ecommerce>. Acesso em: 19 jun. 2025.

DIO. **DQL (Data Query Language)**. Disponível em: <https://www.dio.me/articles/dql-data-query-language>. Acesso em: 19 jun. 2025.

VER. **DTL – Linguagem de Transação de Dados**. Disponível em: <https://consultabd.wordpress.com/2013/06/21/dtl-linguagem-de-transacao-de-dados/>. Acesso em: 19 jun. 2025.

REIS, F. DOS. **Comandos DML SQL e sua sintaxe**. Disponível em: <https://www.bosontreinamentos.com.br/bancos-de-dados/comandos-dml-sql-e-sua-sintaxe/>. Acesso em: 19 jun. 2025.

Comandos SQL DDL: O guia definitivo. Disponível em: <https://www.datacamp.com/pt/tutorial/sql-ddl-commands>. Acesso em: 19 jun. 2025.

DOS REIS, F. **Comandos DCL SQL e sua sintaxe - Bóson Treinamentos em Ciência e Tecnologia**. Disponível em: <https://www.bosontreinamentos.com.br/bancos-de-dados/comandos-dcl-sql-e-sua-sintaxe/>. Acesso em: 19 jun. 2025.

APÊNDICE A - DDL

```
-- phpMyAdmin SQL Dump

-- versão 5.2.1

-- https://www.phpmyadmin.net/

--

-- Host: 127.0.0.1

-- Tempo de geração: 16/05/2024 às 18:00

-- Versão do servidor: 10.4.32-MariaDB

-- Versão do PHP: 8.2.12


SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";

SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT
*/;

/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;

/*!40101 SET NAMES utf8mb4 */;
```

```
--

-- Banco de dados: `trabalho_bd`

--


-- --------------------------------------------------------

-- Cuidado isso apaga o banco de dados existente!

-- --------------------------------------------------------

DROP DATABASE IF EXISTS `trabalho_bd`;

CREATE DATABASE `trabalho_bd`;

USE `trabalho_bd`;


DELIMITER $$


CREATE PROCEDURE `setup_database`()

BEGIN

   DECLARE EXIT HANDLER FOR SQLEXCEPTION

   BEGIN

      SELECT 'Ocorreu um erro! Revertendo todas as alterações.' AS Mensagem_Erro;

      ROLLBACK;

   END;
```

```
    START TRANSACTION;


-- --------------------------------------------------------


--

-- Estrutura para tabela `Categoria`

--


CREATE TABLE `Categoria` (

  `id` int(11) NOT NULL,

  `nome` varchar(100) NOT NULL,

  `criadoEm` timestamp NOT NULL DEFAULT current_timestamp(),

   `atualizadoEm` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()


) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;


-- --------------------------------------------------------
```

```
--

-- Estrutura para tabela `Usuario`

--


CREATE TABLE `Usuario` (

  `id` int(11) NOT NULL,

  `primeiroNome` varchar(20) NOT NULL,

  `ultimoNome` varchar(20) NOT NULL,

  `email` varchar(30) NOT NULL,

  `senha` varchar(400) NOT NULL,

  `verificado` tinyint(1) NOT NULL,

  `ativo` tinyint(1) NOT NULL,

  `funcionario` tinyint(1) NOT NULL,

  `ultimoLogin` datetime DEFAULT current_timestamp(),

  `superUsuario` tinyint(1) NOT NULL,

  `criadoEm` timestamp NOT NULL DEFAULT current_timestamp(),

   `atualizadoEm` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()


) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
-- --------------------------------------------------------

--

-- Estrutura para tabela `Produto`

--

CREATE TABLE `Produto` (
  `id` int(11) NOT NULL,
  `nome` varchar(40) NOT NULL,
  `categoriaId` int(11) DEFAULT NULL,
  `subCategoria` varchar(40) NOT NULL,
  `descricao` varchar(150) NOT NULL,
  `nomeUrl` varchar(40) DEFAULT NULL,
  `estoque` int(100) NOT NULL,
  `preco` double NOT NULL,
  `promocao` tinyint(1) NOT NULL,
  `criadoEm` timestamp NOT NULL DEFAULT current_timestamp(),
   `atualizadoEm` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;


-- --------------------------------------------------------


--

-- Estrutura para tabela `FotoProduto`

--


CREATE TABLE `FotoProduto` (

  `id` int(11) NOT NULL,

  `produtoId` int(11) NOT NULL,

  `principal` tinyint(1) NOT NULL,

  `caminho` char(255) NOT NULL,

  `criadoEm` timestamp NOT NULL DEFAULT current_timestamp(),

   `atualizadoEm` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;


-- --------------------------------------------------------
```

```
--

-- Estrutura para tabela `Pedido`

--


CREATE TABLE `Pedido` (

  `id` int(11) NOT NULL,

  `status` varchar(50) NOT NULL,

  `usuarioId` int(11) DEFAULT NULL,

  `valorTotal` float NOT NULL,

  `criadoEm` timestamp NOT NULL DEFAULT current_timestamp(),

   `atualizadoEm` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()


) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;


-- --------------------------------------------------------


--

-- Estrutura para tabela `ItemPedido`
```

```
--

CREATE TABLE `ItemPedido` (
  `id` int(11) NOT NULL,
  `pedidoId` int(11) NOT NULL,
  `produtoId` int(11) NOT NULL,
  `quantidade` int(11) NOT NULL,
  `criadoEm` timestamp NOT NULL DEFAULT current_timestamp(),
   `atualizadoEm` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- --------------------------------------------------------

--
-- Estrutura para tabela `Carrinho`
--

CREATE TABLE `Carrinho` (
```

```
  `id` int(11) NOT NULL,

  `produtoId` int(11) NOT NULL,

  `usuarioId` int(11) NOT NULL,

  `criadoEm` timestamp NOT NULL DEFAULT current_timestamp(),

   `atualizadoEm` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;



-- --------------------------------------------------------



--

-- Estrutura para tabela `ItemCarrinho`

--



CREATE TABLE `ItemCarrinho` (

  `id` int(11) NOT NULL,

  `carrinhoId` int(11) NOT NULL,

  `produtoId` int(11) NOT NULL,

  `quantidade` int(11) NOT NULL,

  `precoInd` double(10,0) NOT NULL,
```

```
  `criadoEm` timestamp NOT NULL DEFAULT current_timestamp(),
  `atualizadoEm` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Índices para tabelas despejadas
--

--
-- Índices de tabela `Categoria`
--
ALTER TABLE `Categoria`
  ADD PRIMARY KEY (`id`);

--
-- Índices de tabela `Usuario`
--
ALTER TABLE `Usuario`
  ADD PRIMARY KEY (`id`);
```

```
--
-- Índices de tabela `Produto`
--
ALTER TABLE `Produto`
  ADD PRIMARY KEY (`id`),
  ADD KEY `categoriaId` (`categoriaId`);


--
-- Índices de tabela `FotoProduto`
--
ALTER TABLE `FotoProduto`
  ADD PRIMARY KEY (`id`),
  ADD KEY `produtoId` (`produtoId`);


--
-- Índices de tabela `Pedido`
--
ALTER TABLE `Pedido`
  ADD PRIMARY KEY (`id`),
```

```
  ADD KEY `usuarioId` (`usuarioId`);


--

-- Índices de tabela `ItemPedido`

--

ALTER TABLE `ItemPedido`

  ADD PRIMARY KEY (`id`),

  ADD KEY `produtoId` (`produtoId`),

  ADD KEY `pedidoId` (`pedidoId`);


--

-- Índices de tabela `Carrinho`

--

ALTER TABLE `Carrinho`

  ADD PRIMARY KEY (`id`),

  ADD KEY `produtoId` (`produtoId`),

  ADD KEY `usuarioId` (`usuarioId`);


--

-- Índices de tabela `ItemCarrinho`
```

```
--
ALTER TABLE `ItemCarrinho`
  ADD PRIMARY KEY (`id`),
  ADD KEY `carrinhoId` (`carrinhoId`),
  ADD KEY `produtoId` (`produtoId`);


--
-- AUTO_INCREMENT para tabelas despejadas
--


--
-- AUTO_INCREMENT de tabela `Categoria`
--
ALTER TABLE `Categoria`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;


--
-- AUTO_INCREMENT de tabela `Usuario`
--
ALTER TABLE `Usuario`
```

```
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=34;


--

-- AUTO_INCREMENT de tabela `Produto`

--

ALTER TABLE `Produto`

  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=27;


--

-- AUTO_INCREMENT de tabela `FotoProduto`

--

ALTER TABLE `FotoProduto`

  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;


--

-- AUTO_INCREMENT de tabela `Pedido`

--

ALTER TABLE `Pedido`

  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=159;
```

```
--

-- AUTO_INCREMENT de tabela `ItemPedido`

--

ALTER TABLE `ItemPedido`

  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=45;



--

-- AUTO_INCREMENT de tabela `Carrinho`

--

ALTER TABLE `Carrinho`

  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;



--

-- AUTO_INCREMENT de tabela `ItemCarrinho`

--

ALTER TABLE `ItemCarrinho`

  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=56;



--

-- Restrições para tabelas despejadas
```

```
--


--

-- Restrições para tabela `Produto`

--

ALTER TABLE `Produto`

  ADD CONSTRAINT `Produto_ibfk_1` FOREIGN KEY (`categoriaId`) REFERENCES `Categoria` (`id`);


--

-- Restrições para tabela `FotoProduto`

--

ALTER TABLE `FotoProduto`

 ADD CONSTRAINT `FotoProduto_ibfk_1` FOREIGN KEY (`produtoId`) REFERENCES `Produto` (`id`);


--

-- Restrições para tabela `Pedido`

--

ALTER TABLE `Pedido`
```

```
  ADD CONSTRAINT `Pedido_ibfk_1` FOREIGN KEY (`usuarioId`) REFERENCES `Usuario` (`id`);


--

-- Restrições para tabela `ItemPedido`

--

ALTER TABLE `ItemPedido`

  ADD CONSTRAINT `ItemPedido_ibfk_1` FOREIGN KEY (`pedidoId`) REFERENCES `Pedido` (`id`),

  ADD CONSTRAINT `ItemPedido_ibfk_2` FOREIGN KEY (`produtoId`) REFERENCES `Produto` (`id`);


--

-- Restrições para tabela `Carrinho`

--

ALTER TABLE `Carrinho`

  ADD CONSTRAINT `Carrinho_ibfk_1` FOREIGN KEY (`produtoId`) REFERENCES `Produto` (`id`),

  ADD CONSTRAINT `Carrinho_ibfk_2` FOREIGN KEY (`usuarioId`) REFERENCES `Usuario` (`id`);


--
```

```
-- Restrições para tabela `ItemCarrinho`

--

ALTER TABLE `ItemCarrinho`

    ADD CONSTRAINT `ItemCarrinho_ibfk_1` FOREIGN KEY (`carrinhoId`) REFERENCES `Carrinho` (`id`),

    ADD CONSTRAINT `ItemCarrinho_ibfk_2` FOREIGN KEY (`produtoId`) REFERENCES `Produto` (`id`);


  COMMIT;


END$$

DELIMITER ;


CALL `setup_database`();


DROP PROCEDURE IF EXISTS `setup_database`;


/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

APÊNDICE B - INSERÇÃO DE DADOS (DML)

```
USE `trabalho_bd`;

DELIMITER $$


CREATE PROCEDURE realizar_insercoes()

BEGIN

  DECLARE EXIT HANDLER FOR SQLEXCEPTION

  BEGIN

    ROLLBACK;

    SELECT 'Erro na inserção. Transação revertida.' AS Resultado;

  END;


  START TRANSACTION;


  INSERT INTO Categoria (id, nome) VALUES

    (1, 'Eletrônicos'),

    (2, 'Roupas'),

    (3, 'Livros');
```

```
INSERT INTO Usuario (id, primeiroNome, ultimoNome, email, senha, verificado, ativo, funcionario, superUsuario)
VALUES
(1, 'Lucas', 'Silva', 'lucas@email.com', 'senha123', 1, 1, 0, 0),
(2, 'Ana', 'Souza', 'ana@email.com', 'senha456', 1, 1, 1, 0);

-- Atualiza auto_increment para evitar conflito
ALTER TABLE Usuario AUTO_INCREMENT = 3;
ALTER TABLE Categoria AUTO_INCREMENT = 4;

INSERT INTO Produto (id, nome, categoriaId, subCategoria, descricao, nomeUrl, estoque, preco, promocao)
VALUES
(1, 'Smartphone Galaxy', 1, 'Celulares', 'Smartphone Android', 'smartphone-galaxy', 50, 1999.90, 0),
(2, 'Camiseta Branca', 2, 'Masculino', 'Camiseta algodão', 'camiseta-branca', 100, 49.90, 1);

ALTER TABLE Produto AUTO_INCREMENT = 3;

INSERT INTO Pedido (status, usuarioId, valorTotal) VALUES ('Pendente', 1, 2049.80);
```

```
  SET @pedidoId = LAST_INSERT_ID();

  INSERT INTO ItemPedido (pedidoId, produtoId, quantidade)
  VALUES (@pedidoId, 1, 1), (@pedidoId, 2, 1);

  INSERT INTO Carrinho (id, produtoId, usuarioId) VALUES (1, 1, 2);

  INSERT INTO ItemCarrinho (carrinhoId, produtoId, quantidade, precoInd)
  VALUES (1, 1, 2, 1999);

  COMMIT;

  SELECT 'Inserções realizadas com sucesso!' AS Resultado;
END$$

DELIMITER ;

CALL realizar_insercoes();

DROP PROCEDURE IF EXISTS realizar_insercoes;
```

APÊNDICE C - ATUALIZAÇÃO DE DADOS (DML)

```
USE `trabalho_bd`;

DELIMITER $$


CREATE PROCEDURE realizar_atualizacoes()

BEGIN

 DECLARE EXIT HANDLER FOR SQLEXCEPTION

 BEGIN

  ROLLBACK;

  SELECT 'Erro na atualização. Transação revertida.' AS Resultado;

 END;


 START TRANSACTION;


 -- Atualiza nome da categoria com id = 1

 UPDATE Categoria

 SET nome = 'Eletrônicos e Tecnologia'

 WHERE id = 1;


 -- Diminui 1 unidade do estoque do produto com id = 1

 UPDATE Produto

 SET estoque = estoque - 1
```

```
  WHERE id = 1;

  -- Atualiza status do pedido (assumindo o último inserido)
  UPDATE Pedido
  SET status = 'Enviado'
  WHERE usuarioId = 1
  ORDER BY id DESC
  LIMIT 1;

  -- Atualiza ultimoLogin do usuário com id = 1 para o timestamp atual
  UPDATE Usuario
  SET ultimoLogin = NOW()
  WHERE id = 1;

  COMMIT;

  SELECT 'Atualizações realizadas com sucesso!' AS Resultado;
END$$

DELIMITER ;

CALL realizar_atualizacoes();
```

```
DROP PROCEDURE IF EXISTS realizar_atualizacoes;
```

APÊNDICE D - DELEÇÃO DE DADOS (DML)

```
USE `trabalho_bd`;

DELIMITER $$


CREATE PROCEDURE realizar_delecoes()

BEGIN

  DECLARE ultimoPedidoId INT;


  DECLARE EXIT HANDLER FOR SQLEXCEPTION

  BEGIN

    ROLLBACK;

    SELECT 'Erro na exclusão. Transação revertida.' AS Resultado;

  END;


  START TRANSACTION;


  -- Busca o último pedido do usuário 1

   SELECT id INTO ultimoPedidoId FROM Pedido WHERE usuarioId = 1 ORDER BY id
DESC LIMIT 1;


  DELETE FROM ItemCarrinho WHERE carrinhoId = 1;


  DELETE FROM Carrinho WHERE id = 1;
```

```
  DELETE FROM ItemPedido WHERE pedidoId = ultimoPedidoId;

  DELETE FROM Pedido WHERE id = ultimoPedidoId;

  DELETE FROM Produto WHERE id IN (1, 2);

  DELETE FROM Usuario WHERE id IN (1, 2);

  DELETE FROM Categoria WHERE id IN (1, 2, 3);

  COMMIT;

  SELECT 'Deleções realizadas com sucesso!' AS Resultado;
END$$

DELIMITER ;

CALL realizar_delecoes();
DROP PROCEDURE IF EXISTS realizar_delecoes;
```

APÊNDICE E - DQL

```
-- Fazer uma consulta que retorne a quantidade de pedidos por status.

SELECT status, COUNT(*) AS total

FROM Pedido

GROUP BY status;


-- Fazer uma consulta que retorne o total vendido por produto, ordenado por produtos mais vendidos.

SELECT p.nome, SUM(ip.quantidade) AS total_vendido

FROM ItemPedido ip

JOIN Produto p ON ip.produtoId = p.id

GROUP BY p.nome

ORDER BY total_vendido DESC

LIMIT 10;


-- Fazer uma consulta que retorne o total gasto por usuário, ordenado do maior para o menor.

SELECT u.primeiroNome, u.ultimoNome, COUNT(p.id) AS total_pedidos,

    SUM(p.valorTotal) AS total_gasto

FROM Pedido p

JOIN Usuario u ON p.usuarioId = u.id

GROUP BY u.id

ORDER BY total_gasto DESC;
```

```
-- Fazer uma consulta que retorne o nome e preço dos produtos em promoção com estoque abaixo de 10.

SELECT nome, preco, estoque

FROM Produto

WHERE promocao = 1 AND estoque < 10;


-- Fazer uma consulta que retorne o total de produtos por categoria.

SELECT c.nome AS categoria, COUNT(p.id) AS total_produtos

FROM Produto p

JOIN Categoria c ON p.categoriaId = c.id

GROUP BY c.nome;


-- Fazer uma consulta que retorne o total de itens no carrinho por usuário, ordenado do maior para o menor.

SELECT u.email, COUNT(ic.id) AS total_itens

FROM ItemCarrinho ic

JOIN Carrinho c ON ic.carrinhoId = c.id

JOIN Usuario u ON c.usuarioId = u.id

GROUP BY u.id

ORDER BY total_itens DESC;
```

APÊNDICE F - DCL

```
USE `trabalho_bd`;

DROP USER IF EXISTS 'administrador'@'localhost';
DROP USER IF EXISTS 'cliente_01'@'localhost';
DROP USER IF EXISTS 'cliente_02'@'localhost';
DROP USER IF EXISTS 'cliente_03'@'localhost';

CREATE USER 'administrador'@'localhost' IDENTIFIED BY 'senha_supersegura195';
CREATE USER 'cliente_01'@'localhost' IDENTIFIED BY 'senha_390';
CREATE USER 'cliente_02'@'localhost' IDENTIFIED BY 'senha_821';
CREATE USER 'cliente_03'@'localhost' IDENTIFIED BY 'senha_093';

GRANT ALL PRIVILEGES ON trabalho_bd.* TO 'administrador'@'localhost';

GRANT ALL PRIVILEGES ON trabalho_bd.* TO 'cliente_01'@'localhost';
REVOKE INSERT, UPDATE, DELETE ON trabalho_bd.* FROM 'cliente_01'@'localhost';

GRANT SELECT ON trabalho_bd.* TO 'cliente_02'@'localhost';

GRANT UPDATE ON trabalho_bd.Produto TO 'cliente_02'@'localhost';
GRANT UPDATE ON trabalho_bd.FotoProduto TO 'cliente_02'@'localhost';
```

```
GRANT UPDATE ON trabalho_bd.Carrinho TO 'cliente_02'@'localhost';

GRANT UPDATE ON trabalho_bd.Pedido TO 'cliente_02'@'localhost';

GRANT UPDATE ON trabalho_bd.ItemPedido TO 'cliente_02'@'localhost';


GRANT ALL PRIVILEGES ON trabalho_bd.* TO 'cliente_03'@'localhost';

REVOKE DELETE ON trabalho_bd.* FROM 'cliente_03'@'localhost';


FLUSH PRIVILEGES;
```