IV - DADOS E BASES DE DADOS

1. Conceitos fundamentais de dados: o que são dados; processos geradores de dados; tipos e classes de dados; formatos de arquivos de dados comuns (txt, csv, xlsx, xml, json e parquet).

1.1 O que são dados? 📊

Dados são representações de informações que podem ser processadas por computadores. Eles podem ser números, textos, imagens, sons ou qualquer forma de input que possa ser interpretado e analisado para gerar conhecimento.

Exemplo: Números em uma planilha de vendas (quantidades, preços, datas) são dados que, quando organizados e analisados, ajudam a entender o desempenho da empresa.

1.2 Processos geradores de dados 🔍

Dados podem ser gerados de várias maneiras, refletindo eventos do mundo real ou atividades computacionais. Os principais processos geradores incluem:

- **Entrada manual**: Inserção de dados por humanos, como o preenchimento de formulários.
- **Sensores e dispositivos IoT**: Captura automática de dados, como temperatura, localização, movimentos, etc.
- **Transações digitais**: Compras online, operações bancárias, registros de login.
- **Log de sistemas**: Registros de eventos e atividades de sistemas operacionais, aplicativos ou redes.
- **Web scraping e APIs**: Extração de dados da internet, coletando informações de sites e serviços online.

Exemplo: Um sistema de e-commerce gera dados a partir das transações dos clientes, incluindo informações como produtos comprados, quantidade, tempo de compra e método de pagamento.

1.3 Tipos e classes de dados 📦

Os dados podem ser categorizados com base na forma como são estruturados e na sua natureza:

- **Dados Estruturados**: Dados organizados em formatos predefinidos, como tabelas com linhas e colunas (ex.: banco de dados SQL).
    - **Exemplo**: Tabelas de um banco de dados relacional com colunas para nome, idade e cidade.
- **Dados Semiestruturados**: Têm uma estrutura parcial, mas não são rígidos como os estruturados. São comuns em formatos como XML e JSON.
    - **Exemplo**: Um documento JSON que armazena dados de um usuário em pares chave-valor.
- **Dados Não Estruturados**: Dados sem uma estrutura definida, como imagens, vídeos, e-mails, documentos de texto, etc.
    - **Exemplo**: Fotografias e vídeos armazenados em um servidor.

- **Dados Qualitativos**: Dados descritivos que não são numéricos, como opiniões ou feedback de clientes.
    - **Exemplo**: Comentários deixados por clientes sobre produtos.
- **Dados Quantitativos**: Dados que podem ser medidos e expressos numericamente.
    - **Exemplo**: Números de vendas diárias ou medições de temperatura.

1.4 Formatos de arquivos de dados comuns 📂

- **TXT (Text File)**: Formato básico que armazena dados como texto simples. É amplamente usado para registros de log e armazenamento de pequenas quantidades de dados.
    - **Exemplo**: Arquivos de log de servidores onde cada linha é um registro de evento.
- **CSV (Comma-Separated Values)**: Formato tabular onde os valores são separados por vírgulas. Muito utilizado para exportação e importação de dados em tabelas.
    - **Exemplo**: Uma planilha de Excel exportada como CSV contendo informações de vendas.
- **XLSX (Excel File)**: Formato proprietário do Microsoft Excel que armazena dados em planilhas com múltiplas abas, fórmulas e gráficos. Usado para manipulação e análise de dados tabulares.
    - **Exemplo**: Relatórios financeiros que utilizam fórmulas complexas para calcular lucros e perdas.
- **XML (eXtensible Markup Language)**: Formato que armazena dados em uma estrutura hierárquica, sendo útil para transferir dados entre sistemas de forma legível tanto para humanos quanto para máquinas.
    - **Exemplo**: Arquivos de configuração de software ou troca de dados entre aplicativos (ex.: sistemas de reservas de passagens aéreas).
- **JSON (JavaScript Object Notation)**: Formato leve para troca de dados, com estrutura de chave-valor que facilita a leitura e escrita por humanos e sistemas. Muito utilizado em APIs e aplicações web.
    - **Exemplo**: Respostas de APIs que retornam informações do usuário, como nome, idade e endereço em formato JSON.
- **Parquet**: Formato de armazenamento otimizado para leitura e compressão, usado principalmente em Big Data para armazenar dados analíticos. É colunar, permitindo consultas rápidas e eficientes.
    - **Exemplo**: Dados de log de grandes volumes processados por ferramentas como Apache Spark ou Hadoop.

Resumo das Possíveis Cobranças em Provas:

- **Definição de Dados**: Pode ser cobrada a definição e exemplos práticos de como dados são usados.
- **Tipos de Dados**: Questões podem focar nas diferenças entre dados estruturados, semiestruturados e não estruturados, com exemplos.

- **Processos Geradores de Dados**: Questões podem pedir que você identifique processos que geram dados (ex.: sensores IoT) e como esses dados são usados.
- **Formatos de Arquivos**: É comum que perguntas abordem os formatos de dados (ex.: diferença entre JSON e CSV) e seus usos apropriados em diferentes cenários.

2. Introdução a Bases de Dados: o que são bases de dados; tipos de bases de dados; metadados; tidy data.

2.1 O que são Bases de Dados?

Bases de dados são coleções organizadas de informações que permitem o armazenamento, recuperação e manipulação eficiente dos dados. Elas são essenciais para gerenciar grandes quantidades de informações de forma estruturada e acessível.

- **Exemplo**: Um sistema de gestão de clientes (CRM) que armazena informações de contato, histórico de compras e interações de cada cliente em um banco de dados relacional.

2.2 Tipos de Bases de Dados

- **Bases de Dados Relacionais**: Estruturadas em tabelas com linhas e colunas, onde os dados são organizados em registros e os relacionamentos são definidos por chaves. Utilizam SQL (Structured Query Language) para manipulação de dados.
    - **Exemplo**: MySQL, PostgreSQL, Oracle.
- **Bases de Dados NoSQL**: Projetadas para lidar com grandes volumes de dados não estruturados ou semiestruturados, como documentos, grafos ou chave-valor, oferecendo flexibilidade e escalabilidade.
    - **Subtipos**:
        - **Chave-Valor**: Armazena dados em pares chave-valor. Exemplo: Redis.
        - **Documentos**: Armazena dados como documentos JSON ou BSON. Exemplo: MongoDB.
        - **Grafos**: Otimizadas para explorar relações entre dados. Exemplo: Neo4j.
        - **Colunares**: Armazenam dados por colunas, otimizadas para leitura rápida. Exemplo: Apache Cassandra.
- **Bases de Dados Distribuídas**: Armazenam dados em múltiplos servidores ou locais, distribuindo a carga de processamento e aumentando a disponibilidade.
    - **Exemplo**: Google Bigtable, Amazon DynamoDB.
- **Data Warehouses**: Projetadas para análise de grandes volumes de dados históricos. Focam na otimização de consultas complexas.
    - **Exemplo**: Amazon Redshift, Snowflake.

2.3 Metadados

Metadados são "dados sobre dados". Eles descrevem as propriedades e características dos dados, como tipo, formato, estrutura e restrições, ajudando a organizar, entender e gerenciar as bases de dados.

- **Exemplo**: Em uma tabela de um banco de dados, metadados incluem informações como o nome da tabela, nomes das colunas, tipos de dados das colunas (ex.: inteiro, texto) e restrições (ex.: chave primária, chave estrangeira).

- **Funções dos Metadados**:

    - Facilitam a gestão e busca de informações dentro de uma base de dados.
    - Descrevem a estrutura dos dados para garantir consistência e integridade.
    - Ajudam na documentação e compreensão dos dados por usuários e sistemas.

2.4 Tidy Data

"Tidy data" refere-se a um formato padrão para organizar dados de forma limpa e estruturada, facilitando a análise. Em tidy data, cada variável forma uma coluna, cada observação forma uma linha, e cada tipo de unidade observacional forma uma tabela.

- **Princípios de Tidy Data**:

    - Cada coluna representa uma variável.
    - Cada linha representa uma observação.
    - Cada tabela armazena um tipo de unidade observacional.

- **Exemplo**:

Nome	Idade	Cidade
João	25	São Paulo
Maria	30	Rio

Neste exemplo, a tabela segue o formato tidy data, onde cada linha representa uma pessoa, e cada coluna representa uma característica (variável) dessa pessoa.

Resumo das Possíveis Cobranças em Provas:

- **Definição de Bases de Dados**: Questões podem abordar o conceito de bases de dados e sua importância na organização de informações.
- **Tipos de Bases de Dados**: Questões podem diferenciar entre bases de dados relacionais, NoSQL, distribuídas e data warehouses, incluindo suas aplicações.
- **Metadados**: Podem ser cobradas as funções e exemplos de metadados dentro de uma base de dados.
- **Tidy Data**: Podem ser abordados os princípios do tidy data e sua importância na análise de dados.

3. Introdução ao Armazenamento de Dados: armazenamento de arquivos; principais estruturas de armazenamento de dados analíticos (data warehouse, data mart, data lake, data lakehouse, vector stores), suas diferenças conceituais e casos de uso; armazenamento na nuvem.

3.1 Armazenamento de Arquivos

Armazenamento de arquivos refere-se à prática de guardar dados em sistemas de arquivos tradicionais, onde os dados são organizados em pastas e subpastas. É um método comum para armazenar documentos, imagens, vídeos, e outros tipos de arquivos não estruturados.

- **Exemplo**: Armazenar documentos em um servidor de arquivos da empresa ou em um serviço de armazenamento em nuvem como Google Drive ou Dropbox.

3.2 Estruturas de Armazenamento de Dados Analíticos

3.2.1 Data Warehouse

- **Definição**: Um data warehouse é um sistema de armazenamento de dados estruturados projetado para análise e relatórios. Ele consolida dados de diferentes fontes e os organiza de forma otimizada para consultas complexas e rápidas.
- **Características**:
    - Armazenamento de dados históricos.
    - Estruturado para análises e relatórios.
    - Alta performance em consultas complexas.
- **Caso de Uso**: Utilizado por empresas para gerar relatórios financeiros, análises de vendas e outras métricas de negócios.
- **Exemplo**: Amazon Redshift, Google BigQuery.

3.2.2 Data Mart

- **Definição**: Um data mart é uma versão menor e mais focada de um data warehouse, destinada a atender as necessidades específicas de um departamento ou área de negócios.
- **Características**:
    - Segmento de um data warehouse.
    - Focado em um conjunto específico de dados (ex.: vendas, marketing).
    - Rápido e direcionado para usuários finais.
- **Caso de Uso**: Departamentos de marketing usam data marts para acessar dados relevantes para campanhas e análise de clientes.
- **Exemplo**: Data mart focado em vendas dentro de um data warehouse maior.

3.2.3 Data Lake

- **Definição**: Um data lake é um repositório que permite armazenar grandes volumes de dados brutos em seu formato original, sejam eles estruturados, semiestruturados ou não estruturados.
- **Características**:
    - Armazena dados em sua forma bruta.
    - Suporta diversos formatos de dados (JSON, XML, CSV, etc.).

    - Flexível e escalável.

- **Caso de Uso**: Usado para ingestão e armazenamento de grandes volumes de dados que podem ser analisados posteriormente com técnicas de machine learning ou big data.
- **Exemplo**: Azure Data Lake, Amazon S3.

3.2.4 Data Lakehouse

- **Definição**: Um data lakehouse combina as vantagens do data warehouse e do data lake, oferecendo uma estrutura unificada que permite o armazenamento de dados brutos e a execução de análises estruturadas.
- **Características**:
    - Combina a flexibilidade do data lake com o desempenho do data warehouse.
    - Suporta dados estruturados e não estruturados.
    - Facilita análises em tempo real e batch.
- **Caso de Uso**: Empresas que necessitam tanto de armazenamento de dados brutos para aprendizado de máquina quanto de dados estruturados para relatórios rápidos.
- **Exemplo**: Databricks Lakehouse, Google BigLake.

3.2.5 Vector Stores

- **Definição**: Armazenamento de vetores é utilizado para armazenar representações vetoriais de dados, como embeddings gerados por modelos de aprendizado de máquina para tarefas como busca semântica, recomendação de produtos e análise de sentimentos.
- **Características**:
    - Armazena dados na forma de vetores.
    - Otimizado para buscas rápidas e comparações de similaridade.
    - Usado em inteligência artificial e aprendizado de máquina.
- **Caso de Uso**: Empresas que usam modelos de IA para recomendação de produtos, buscas semânticas e outros algoritmos de similaridade.
- **Exemplo**: Pinecone, Milvus.

3.3 Armazenamento na Nuvem

- **Definição**: Armazenamento na nuvem refere-se ao uso de serviços fornecidos por provedores de nuvem para armazenar dados remotamente, acessíveis via internet. Oferece escalabilidade, flexibilidade e acessibilidade global.
- **Principais Provedores**:
    - **Amazon Web Services (AWS)**: S3, EBS, Glacier.
    - **Microsoft Azure**: Blob Storage, Azure Data Lake.
    - **Google Cloud**: Google Cloud Storage, BigQuery.

- **Vantagens**:
    - Escalabilidade quase ilimitada.
    - Pagamento conforme o uso.
    - Redundância e alta disponibilidade.
- **Caso de Uso**: Empresas de todos os tamanhos usam armazenamento na nuvem para backups, hospedagem de sites, armazenamento de grandes volumes de dados e análise de big data.

Resumo das Possíveis Cobranças em Provas:

- **Armazenamento de Arquivos**: Questões podem abordar conceitos básicos de armazenamento de arquivos e sua importância.
- **Estruturas de Armazenamento Analítico**: Podem ser cobradas as diferenças entre data warehouse, data mart, data lake, data lakehouse e vector stores, além de seus casos de uso.
- **Armazenamento na Nuvem**: Perguntas podem focar nas vantagens, principais provedores e usos comuns do armazenamento em nuvem.

4. Sistemas Gerenciadores de Base de Dados (SGBD): definição de SGBD; principais funções; principais tipos de SGBDs (SQL e NoSQL) e suas diferenças; transações e índices.

4.1 Definição de SGBD

Um Sistema Gerenciador de Base de Dados (SGBD) é um software que permite criar, gerenciar, manipular e acessar bases de dados de maneira eficiente e organizada. Ele fornece ferramentas para a definição de estruturas de dados, inserção, atualização, exclusão e consulta dos dados, garantindo segurança, integridade e consistência.

- **Exemplo**: MySQL, PostgreSQL, MongoDB.

4.2 Principais Funções dos SGBDs

- **Criação de Bases de Dados**: Define a estrutura da base de dados, incluindo tabelas, colunas e tipos de dados.
- **Manipulação de Dados**: Permite inserir, atualizar, excluir e consultar dados.
- **Controle de Acesso**: Garante que apenas usuários autorizados possam acessar ou modificar os dados.
- **Segurança e Integridade**: Protege os dados contra acessos não autorizados e mantém a integridade das informações armazenadas.
- **Gerenciamento de Transações**: Permite que operações sejam executadas de forma atômica, consistente, isolada e durável (ACID).
- **Recuperação de Falhas**: Possui mecanismos para restaurar a base de dados após falhas, mantendo a consistência.

4.3 Principais Tipos de SGBDs (SQL e NoSQL)

4.3.1 SGBDs SQL (Relacionais)

- **Definição**: SGBDs que utilizam uma linguagem estruturada de consulta (SQL) para manipular dados em bases de dados relacionais, onde os dados são organizados em tabelas com colunas e linhas.
- **Características**:
    - Estrutura de dados rígida e bem definida.
    - Relacionamentos entre tabelas usando chaves primárias e estrangeiras.
    - Suporte a transações ACID, garantindo integridade e consistência.
- **Exemplos**: MySQL, PostgreSQL, Oracle, SQL Server.
- **Caso de Uso**: Aplicações que exigem consistência e integridade, como sistemas financeiros, ERPs e CRMs.

4.3.2 SGBDs NoSQL (Não Relacionais)

- **Definição**: SGBDs projetados para lidar com grandes volumes de dados não estruturados ou semiestruturados, oferecendo maior flexibilidade e escalabilidade em comparação aos SGBDs relacionais.
- **Principais Tipos de NoSQL**:
    - **Chave-Valor**: Dados armazenados em pares chave-valor. Exemplo: Redis, DynamoDB.
    - **Documentos**: Armazena dados em documentos JSON ou BSON. Exemplo: MongoDB, CouchDB.
    - **Grafos**: Otimizado para explorar relacionamentos complexos entre dados. Exemplo: Neo4j.
    - **Colunares**: Armazena dados por colunas, otimizando consultas de leitura. Exemplo: Cassandra, HBase.
- **Características**:
    - Flexibilidade de esquema, permitindo mudanças rápidas na estrutura dos dados.
    - Alta escalabilidade horizontal.
    - Desempenho otimizado para grandes volumes de dados e consultas específicas.
- **Caso de Uso**: Aplicações que lidam com grandes volumes de dados, como redes sociais, IoT, e-commerce, e big data.

4.4 Diferenças entre SQL e NoSQL

Característica	SQL	NoSQL
Estrutura	Tabelas, colunas e linhas	Chave-valor, documentos, grafos, colunas
Esquema	Fixo e pré-definido	Flexível e dinâmico
Transações	Suporte completo a ACID	Suporte variável (depende do tipo)
Escalabilidade	Vertical (melhoria de hardware)	Horizontal (adição de servidores)

Característica	SQL	NoSQL
Uso	Aplicações com alta consistência	Aplicações que requerem escalabilidade e flexibilidade

4.5 Transações

- **Definição**: Uma transação é uma sequência de operações que são tratadas como uma única unidade lógica de trabalho. Transações são fundamentais para garantir a integridade dos dados em operações que envolvem múltiplas etapas.
- **Propriedades ACID**:
  - **Atomicidade**: Todas as operações dentro da transação são concluídas ou nenhuma é.
  - **Consistência**: A transação leva o banco de dados de um estado consistente a outro estado consistente.
  - **Isolamento**: Operações de uma transação não são visíveis para outras até serem finalizadas.
  - **Durabilidade**: Uma vez que uma transação é concluída, suas mudanças são permanentes.
- **Exemplo**: Em uma transação bancária, a transferência de fundos entre contas é tratada como uma transação para garantir que os fundos sejam debitados de uma conta e creditados em outra de forma segura.

4.6 Índices

- **Definição**: Índices são estruturas de dados que melhoram a velocidade de recuperação de registros em uma tabela de banco de dados, criando uma referência rápida para localizar dados sem precisar varrer toda a tabela.
- **Tipos de Índices**:
  - **Índice Primário**: Baseado na chave primária da tabela.
  - **Índice Secundário**: Baseado em colunas que não são chaves primárias.
  - **Índices Compostos**: Índices que utilizam múltiplas colunas para melhorar a performance de consultas específicas.
- **Caso de Uso**: Melhorar o desempenho de consultas que buscam dados específicos em grandes tabelas.

5. Modelo de Dados: modelo de entidade-relacionamento (ER); modelo relacional: tabelas, esquemas, chaves, consultas; dados estruturados, semiestruturados e não estruturados; modelo chave-valor; modelo colunar; modelo orientado a documentos; modelo orientado a grafos.

5.1 Modelo de Entidade-Relacionamento (ER)

- **Definição**: O Modelo de Entidade-Relacionamento (ER) é uma abordagem de modelagem de dados que descreve os dados em termos de entidades, atributos e relacionamentos. Ele é usado principalmente para o projeto de bases de dados relacionais.
- **Componentes**:

    - **Entidades**: Representam objetos ou conceitos do mundo real (ex.: Cliente, Produto).
    - **Atributos**: Descrevem as propriedades das entidades (ex.: Nome, Idade).
    - **Relacionamentos**: Definem como as entidades se conectam (ex.: Cliente compra Produto).

- **Diagrama ER**: Visualiza a estrutura da base de dados e como os dados estão interligados.

5.2 Modelo Relacional

- **Definição**: O modelo relacional organiza os dados em tabelas (ou relações), onde cada tabela é composta por linhas (registros) e colunas (atributos). É a base dos SGBDs relacionais.

- **Componentes**:

    - **Tabelas**: Estruturas que armazenam dados organizados em linhas e colunas.
    - **Esquemas**: Estrutura que define a organização do banco de dados, incluindo tabelas, colunas e relacionamentos.
    - **Chaves**:
        - **Chave Primária**: Identifica de forma única cada registro em uma tabela.
        - **Chave Estrangeira**: Cria relacionamentos entre tabelas diferentes.
    - **Consultas**: Realizadas através de SQL para inserir, atualizar, deletar e buscar dados.

- **Exemplo**: Uma tabela de "Clientes" com colunas "ID", "Nome" e "Email".

5.3 Dados Estruturados, Semiestruturados e Não Estruturados

- **Dados Estruturados**: Organizados em um formato predefinido, como tabelas, onde cada dado segue um padrão específico.

    - **Exemplo**: Tabelas em um banco de dados SQL.

- **Dados Semiestruturados**: Possuem alguma estrutura, mas não são tão rígidos quanto os dados estruturados. Comuns em formatos como XML e JSON.

    - **Exemplo**: Arquivos JSON que contêm pares chave-valor.

- **Dados Não Estruturados**: Dados sem uma estrutura fixa, difíceis de categorizar em tabelas.

    - **Exemplo**: E-mails, imagens, vídeos.

5.4 Modelo Chave-Valor

- **Definição**: Armazena dados em pares chave-valor, onde cada chave é única e associada a um valor. É simples e rápido para operações de leitura e escrita.

- **Características**:

    - Alta flexibilidade e escalabilidade.
    - Sem estrutura rígida de dados.

- **Caso de Uso**: Sessões de usuário, caching, e armazenamento de configurações.

- **Exemplo**: Redis, Amazon DynamoDB.

5.5 Modelo Colunar

- **Definição**: Armazena dados por colunas em vez de linhas. Ideal para consultas que envolvem grandes volumes de dados, pois permite acesso rápido a colunas específicas.
- **Características**:
    - Otimizado para leitura de grandes volumes de dados.
    - Suporta compressão eficiente.
- **Caso de Uso**: Análise de grandes volumes de dados em data warehouses.
- **Exemplo**: Apache Cassandra, Google Bigtable.

5.6 Modelo Orientado a Documentos

- **Definição**: Armazena dados como documentos, geralmente em formatos como JSON, BSON ou XML. Cada documento é autônomo, contendo todas as informações necessárias para ser compreendido.
- **Características**:
    - Flexibilidade na estrutura de dados.
    - Ideal para aplicações que lidam com dados não estruturados.
- **Caso de Uso**: Aplicações que exigem rápido desenvolvimento e iteração de modelos de dados.
- **Exemplo**: MongoDB, CouchDB.

5.7 Modelo Orientado a Grafos

- **Definição**: Armazena dados como nós, arestas e propriedades, sendo ideal para representar e consultar relacionamentos complexos entre entidades.
- **Características**:
    - Otimizado para explorar conexões e relacionamentos.
    - Excelente para consultas que envolvem redes e hierarquias.
- **Caso de Uso**: Redes sociais, sistemas de recomendação, detecção de fraudes.
- **Exemplo**: Neo4j, Amazon Neptune.

Resumo das Possíveis Cobranças em Provas:

- **Modelo ER**: Questões podem abordar a criação de diagramas ER e a definição de entidades, atributos e relacionamentos.
- **Modelo Relacional**: Perguntas podem focar na estrutura das tabelas, uso de chaves primárias e estrangeiras, e exemplos de consultas SQL.
- **Dados Estruturados vs. Semiestruturados e Não Estruturados**: Podem ser cobradas as diferenças e exemplos de cada tipo de dado.

- **Modelos de Dados Não Relacionais**: Questões podem explorar as características e casos de uso dos modelos chave-valor, colunar, orientado a documentos e orientado a grafos.

6. Ingestão e Armazenamento de Dados: definição de ingestão em lote (batch) e em tempo real (stream).

6.1 Definição de Ingestão de Dados

Ingestão de dados refere-se ao processo de coletar, importar e carregar dados de várias fontes para um sistema de armazenamento, como bancos de dados, data lakes ou data warehouses, para posterior processamento e análise. A ingestão de dados pode ocorrer de duas formas principais: em lote (batch) e em tempo real (stream).

6.2 Ingestão em Lote (Batch)

- **Definição**: A ingestão em lote envolve a coleta e o processamento de dados em grandes volumes de uma só vez, em intervalos regulares. Os dados são agrupados e processados de forma sequencial, geralmente durante períodos de baixa atividade para minimizar o impacto no desempenho do sistema.

- **Características**:

    - Ideal para processar grandes quantidades de dados de uma vez.
    - Pode ter um atraso significativo entre a coleta dos dados e o processamento.
    - Requer menor capacidade computacional em tempo real, pois o processamento é agendado.
    - Adequado para operações que não exigem atualização imediata dos dados.

- **Casos de Uso**:

    - Processamento de relatórios financeiros diários ou semanais.
    - Análise de dados históricos para identificar tendências.
    - Atualização de sistemas de back-office, como inventários.

- **Exemplo**: Um sistema de varejo que processa todas as vendas do dia à noite, gerando relatórios para análise no dia seguinte.

6.3 Ingestão em Tempo Real (Stream)

- **Definição**: A ingestão em tempo real, ou streaming, envolve o processamento contínuo e instantâneo de dados conforme eles são gerados. Esse método permite que as informações sejam capturadas e analisadas em tempo real, possibilitando ações imediatas.

- **Características**:

    - Processa dados continuamente, com latência mínima.
    - Ideal para aplicações que exigem respostas rápidas e em tempo real.
    - Requer maior capacidade de processamento e infraestrutura para lidar com o fluxo contínuo de dados.
    - Suporta decisões rápidas e automação baseada em eventos.

- **Casos de Uso**:

    - Monitoramento de fraudes em tempo real em transações financeiras.
    - Análise de tráfego de redes sociais para detectar tendências.
    - Sistemas de recomendação em tempo real, como sugerir produtos durante uma navegação de e-commerce.
    - Monitoramento de dispositivos IoT, como sensores de temperatura em ambientes industriais.

- **Exemplo**: Plataformas de streaming de vídeo, como Netflix, que monitoram o comportamento do usuário e ajustam a entrega de conteúdo em tempo real.

Resumo das Possíveis Cobranças em Provas:

- **Definição e Diferenças**: Questões podem abordar a diferença entre ingestão em lote e em tempo real, incluindo características e vantagens de cada método.
- **Casos de Uso**: Perguntas podem focar em identificar qual tipo de ingestão é mais adequado para diferentes cenários de aplicação.
- **Exemplos Práticos**: Pode-se solicitar exemplos de situações em que a ingestão em lote ou em tempo real seria ideal.

7. Big Data: conceito de big data; conceitos gerais sobre técnicas e ferramentas para lidar com grandes volumes de dados (Spark, Hadoop, HDFS e MapReduce).

7.1 Conceito de Big Data

Big Data refere-se a grandes volumes de dados complexos que são gerados a uma alta velocidade e em formatos variados, tornando-os difíceis de gerenciar e processar usando métodos tradicionais. Esses dados são caracterizados pelos "5 Vs":

- **Volume**: Enorme quantidade de dados gerados diariamente.
- **Velocidade**: A rapidez com que os dados são gerados, coletados e processados.
- **Variedade**: Diversidade de tipos de dados (estruturados, semiestruturados e não estruturados).
- **Veracidade**: Qualidade e confiabilidade dos dados.
- **Valor**: O potencial dos dados para gerar insights valiosos e impactar negócios.

Exemplos de Big Data:

- Registros de redes sociais (tweets, posts, curtidas).
- Dados de sensores IoT (temperatura, pressão).
- Logs de servidores e tráfego de websites.

7.2 Técnicas e Ferramentas para Lidar com Big Data

7.2.1 Apache Hadoop

- **Definição**: Hadoop é um framework de código aberto que permite o processamento distribuído de grandes volumes de dados em clusters de servidores utilizando um modelo de programação simplificado.

- **Características**:

- Processamento paralelo de grandes volumes de dados.
- Alta tolerância a falhas através da replicação de dados.
- Escalabilidade horizontal, permitindo a adição de novos nós ao cluster conforme a necessidade.

Componentes Principais do Hadoop**

1. Hadoop Distributed File System (HDFS) - **Armazenamento de Dados**

- **Função**: HDFS é o sistema de arquivos distribuído do Hadoop, responsável por armazenar grandes volumes de dados em clusters de computadores. Ele divide os arquivos em blocos grandes e os distribui por diversos nós para garantir a escalabilidade e a tolerância a falhas.
- **Características**:
  - Altamente escalável e robusto.
  - Ideal para o armazenamento de grandes conjuntos de dados, permitindo leitura e escrita eficientes.

2. MapReduce - **Processamento de Dados**

- **Função**: MapReduce é o modelo de programação do Hadoop que processa dados de forma paralela e distribuída. Ele divide as tarefas em dois passos principais:
  - **Map**: Fase de mapeamento que processa dados de entrada e os transforma em pares de chave-valor.
  - **Reduce**: Fase de redução que agrega os pares de chave-valor para produzir o resultado final.
- **Características**:
  - Focado no processamento em lote.
  - Ideal para grandes volumes de dados que não exigem processamento em tempo real.

7.2.2 Apache Spark

- **Definição**: Spark é um motor de processamento de dados em tempo real, que oferece um modelo de programação flexível e acelera o processamento distribuído de grandes volumes de dados.

- **Características**:

  - Suporta processamento em batch e em tempo real.
  - Usa memória para armazenamento intermediário, o que aumenta a velocidade de processamento comparado ao MapReduce.
  - Oferece bibliotecas para aprendizado de máquina (MLlib), processamento de gráficos (GraphX), e manipulação de dados estruturados (Spark SQL).

- **Exemplo**: Utilizado por empresas de tecnologia para análise em tempo real de dados de transações financeiras e monitoramento de sistemas.

Resumo das Possíveis Cobranças em Provas:

- **Conceito de Big Data**: Questões podem focar nas características dos 5 Vs e em exemplos práticos de Big Data.
- **Ferramentas de Big Data**: Perguntas podem abordar as funções e características de Hadoop, HDFS, MapReduce e Spark, incluindo suas vantagens e casos de uso específicos.

- **Diferenças entre MapReduce e Spark**: Pode ser explorada a eficiência de cada ferramenta e seu impacto na velocidade e flexibilidade do processamento de grandes volumes de dados.

Aspecto	Apache Spark	Apache Hadoop
Processamento	Em memória (rápido, iterativo).	Em disco (mais lento, batch).
Velocidade	Muito rápido, especialmente para análises iterativas.	Mais lento devido ao uso intensivo de disco.
Facilidade de Uso	APIs amigáveis (Python, Scala, Java, R).	Baseado em Java; mais complexo.
Armazenamento	Depende de HDFS ou outros sistemas.	HDFS (sistema de arquivos nativo).
Iteração	Ideal para algoritmos iterativos.	Iterações custosas e lentas.
Tolerância a Falhas	Alta, com RDDs.	Alta, com replicação no HDFS.
Streaming	Suporta streaming (Spark Streaming).	Não é nativo, mas integra-se com outras ferramentas.
Escalabilidade	Altamente escalável para Big Data.	Extremamente escalável.
Casos de Uso	Aprendizado de máquina, ETL, análises em tempo real.	Processamento de batch, grandes arquivos.
Complexidade	Requer ajustes de memória e configuração.	Simples, mas menos eficiente sem otimizações.

Questoes discursivas

Questão 1

Qual dos seguintes componentes do Apache Hadoop é responsável por armazenar grandes conjuntos de dados de forma distribuída?

- A) MapReduce
- B) Spark
- C) HDFS
- D) Hive

▶ **Resposta: C) HDFS**

Explicação:

- **C) HDFS (Hadoop Distributed File System)** é o sistema de arquivos distribuído do Hadoop que armazena grandes volumes de dados de forma redundante em clusters, garantindo alta disponibilidade.
- **A) MapReduce** é o modelo de programação do Hadoop para processamento de dados, mas não é responsável pelo armazenamento.

- **B) Spark** é um motor de processamento paralelo mais rápido que o MapReduce, mas não é usado para armazenamento de dados.
- **D) Hive** é uma ferramenta para consulta de dados no Hadoop usando SQL-like, mas não armazena dados diretamente.

Questão 2

Qual é a principal diferença entre ingestão de dados em lote (batch) e em tempo real (stream)?

- A) Batch processa dados continuamente enquanto Stream processa dados periodicamente.
- B) Batch lida com grandes volumes de dados em intervalos regulares, enquanto Stream processa dados conforme são gerados.
- C) Batch é usado para processamento em tempo real e Stream para processamento offline.
- D) Batch é mais rápido que Stream.

▶ **Resposta: B) Batch lida com grandes volumes de dados em intervalos regulares, enquanto Stream processa dados conforme são gerados.**

Explicação:

- **B) Correto**: Batch processa grandes volumes de dados em intervalos específicos, enquanto Stream processa dados continuamente à medida que são gerados.
- **A) Errado**: É o oposto; Stream é o que processa continuamente.
- **C) Errado**: Batch não é usado para tempo real, mas para processamentos periódicos.
- **D) Errado**: Stream geralmente oferece processamento mais rápido por ser contínuo, enquanto Batch pode ter um atraso.

Questão 3

Qual das seguintes características é típica de um modelo de dados orientado a grafos?

- A) Armazenamento de dados em pares chave-valor.
- B) Estruturação de dados em tabelas com colunas e linhas.
- C) Representação de dados como nós, arestas e propriedades.
- D) Organização de dados por colunas para leitura rápida.

▶ **Resposta: C) Representação de dados como nós, arestas e propriedades.**

Explicação:

- **C) Correto**: O modelo de grafos é ideal para representar conexões complexas, como redes sociais.
- **A) Errado**: Chave-valor é característico do modelo chave-valor.
- **B) Errado**: Estruturas de tabelas são típicas de modelos relacionais.
- **D) Errado**: A organização por colunas refere-se ao modelo colunar.

Questão 4

O que define um Data Lake em comparação a um Data Warehouse?

- A) Armazena apenas dados estruturados e altamente organizados.
- B) Armazena dados em sua forma bruta, suportando diversos formatos.
- C) É usado exclusivamente para relatórios financeiros.
- D) Tem capacidade limitada para escalabilidade.

▶ **Resposta: B) Armazena dados em sua forma bruta, suportando diversos formatos.**

Explicação:

- **B) Correto**: Data Lakes armazenam dados brutos, sejam eles estruturados, semiestruturados ou não estruturados.
- **A) Errado**: Esta é uma característica de Data Warehouses, que organizam dados de forma estruturada.
- **C) Errado**: Data Warehouses são mais usados para relatórios financeiros.
- **D) Errado**: Data Lakes são altamente escaláveis, ao contrário da opção sugerida.

Questão 5

Qual é o principal benefício de usar Apache Spark em comparação com MapReduce?

- A) Menor custo de implementação.
- B) Suporte exclusivo para dados estruturados.
- C) Processamento mais rápido devido ao uso de memória.
- D) Melhor integração com SQL.

▶ **Resposta: C) Processamento mais rápido devido ao uso de memória.**

Explicação:

- **C) Correto**: Spark armazena dados na memória durante o processamento, o que o torna significativamente mais rápido que o MapReduce, que usa disco para armazenar dados temporários.
- **A) Errado**: O custo de implementação depende do contexto, mas não é um benefício específico do Spark.
- **B) Errado**: Spark suporta tanto dados estruturados quanto não estruturados.
- **D) Errado**: Ambos suportam SQL, mas a principal vantagem do Spark é sua velocidade.

Questão 6

Qual tipo de SGBD é mais adequado para aplicações que exigem alta escalabilidade horizontal e flexibilidade de esquema?

- A) SGBDs Relacionais (SQL)
- B) SGBDs NoSQL
- C) SGBDs Hierárquicos
- D) SGBDs de Redes

▶ **Resposta: B) SGBDs NoSQL**

Explicação:

- **B) Correto**: NoSQL é projetado para alta escalabilidade horizontal e flexibilidade de esquema, sendo ideal para aplicações modernas que lidam com grandes volumes de dados dinâmicos.
- **A) Errado**: SGBDs Relacionais são menos escaláveis horizontalmente e possuem esquemas rígidos.
- **C) Errado**: SGBDs Hierárquicos são antigos e não são usados para alta escalabilidade.
- **D) Errado**: SGBDs de Redes são raramente usados atualmente e não oferecem a escalabilidade do NoSQL.

Questão 7

Em um modelo colunar, qual é a principal vantagem em relação aos modelos tradicionais de tabelas?

- A) Menor necessidade de armazenamento de dados.
- B) Execução de consultas que envolvem múltiplas colunas é mais lenta.
- C) Armazenamento de dados por colunas permite leituras rápidas de colunas específicas.
- D) Melhor para operações de escrita do que leitura.

▶ **Resposta: C) Armazenamento de dados por colunas permite leituras rápidas de colunas específicas.**

Explicação:

- **C) Correto**: Modelos colunares são otimizados para leituras rápidas de colunas específicas, o que acelera as consultas analíticas.
- **A) Errado**: Embora seja eficiente, a redução do armazenamento depende de técnicas como compressão.
- **B) Errado**: Consultas que envolvem múltiplas colunas podem ser otimizadas, mas o foco principal é na leitura de colunas isoladas.
- **D) Errado**: Modelos colunares são mais eficientes para leitura, não escrita.

Questão 8

Qual das seguintes opções melhor descreve um Data Mart?

- A) Um repositório de dados brutos e não processados.
- B) Uma versão focada e segmentada de um Data Warehouse para atender áreas específicas.
- C) Um sistema de arquivos distribuídos para grandes volumes de dados.
- D) Uma estrutura para análise de dados em tempo real.

▶ **Resposta: B) Uma versão focada e segmentada de um Data Warehouse para atender áreas específicas.**

Explicação:

- **B) Correto**: Data Marts são pequenas partes de um Data Warehouse, focadas em necessidades específicas de departamentos, como vendas ou marketing.
- **A) Errado**: Essa é uma descrição de um Data Lake.
- **C) Errado**: Isso descreve HDFS.
- **D) Errado**: Análises em tempo real são mais características de sistemas de streaming.

Questão 9

Qual modelo de dados armazena informações como pares chave-valor?

- A) Modelo Relacional
- B) Modelo Colunar
- C) Modelo Orientado a Documentos
- D) Modelo Chave-Valor

▶ **Resposta: D) Modelo Chave-Valor**

Explicação:

- **D) Correto**: Este modelo armazena dados em pares chave-valor, sendo simples e rápido para operações de leitura e escrita.
- **A) Errado**: O modelo relacional usa tabelas e chaves para relacionar dados.
- **B) Errado**: Modelos colunares armazenam dados em colunas.
- **C) Errado**: Modelos orientados a documentos usam estruturas como JSON ou BSON para armazenar dados.

Questão 10

Qual das seguintes ferramentas é mais adequada para processar grandes volumes de dados em tempo real?

- A) MapReduce
- B) Apache Hive
- C) Apache Spark
- D) HDFS

▶ **Resposta: C) Apache Spark**

Explicação:

- **C) Correto**: Apache Spark é ideal para processamento de dados em tempo real, além de ser mais rápido que MapReduce devido ao uso de memória.
- **A) Errado**: MapReduce é eficiente para processamento em lote, mas não para tempo real.
- **B) Errado**: Hive é usado para consultas SQL-like em Hadoop, não para processamento em tempo real.
- **D) Errado**: HDFS é um sistema de armazenamento e não processa dados diretamente.

Questão 11

Um cientista de dados está projetando um sistema para analisar grandes volumes de dados históricos e gerar relatórios complexos. Qual das seguintes opções de armazenamento seria mais apropriada para esta aplicação e por quê?

- A) Data Mart
- B) Data Warehouse
- C) Data Lake

- D) Vector Store

▶ **Resposta: B) Data Warehouse**

Explicação:

- **B) Correto**: Um Data Warehouse é projetado para armazenar grandes volumes de dados históricos e otimizar consultas complexas, ideal para geração de relatórios.
- **A) Errado**: Data Marts são versões menores de Data Warehouses, focadas em áreas específicas, não para análises complexas e gerais.
- **C) Errado**: Data Lakes armazenam dados em forma bruta e não são otimizados para consultas estruturadas complexas.
- **D) Errado**: Vector Stores são usados principalmente para armazenamento de representações vetoriais, não para análises históricas.

Questão 12

Qual modelo de dados seria mais adequado para um sistema de recomendação que precisa identificar conexões complexas entre diferentes entidades, como produtos e usuários?

- A) Modelo Relacional
- B) Modelo Chave-Valor
- C) Modelo Orientado a Documentos
- D) Modelo Orientado a Grafos

▶ **Resposta: D) Modelo Orientado a Grafos**

Explicação:

- **D) Correto**: O modelo de grafos é ideal para capturar e explorar relações complexas entre entidades, como as que existem em sistemas de recomendação.
- **A) Errado**: O modelo relacional não é eficiente para consultas complexas de conexões entre entidades.
- **B) Errado**: O modelo chave-valor é muito simples para capturar conexões complexas.
- **C) Errado**: O modelo orientado a documentos é mais adequado para armazenar informações sem uma estrutura fixa, mas não para relações complexas.

Questão 13

Uma aplicação de IoT (Internet of Things) coleta dados de sensores a cada segundo e precisa processá-los imediatamente para gerar alertas. Qual método de ingestão de dados é mais adequado para essa aplicação?

- A) Ingestão em Lote (Batch)
- B) Ingestão em Tempo Real (Stream)
- C) Armazenamento em Data Lake
- D) Indexação com MapReduce

▶ **Resposta: B) Ingestão em Tempo Real (Stream)**

Explicação:

- **B) Correto**: Ingestão em tempo real permite processar dados conforme são gerados, essencial para sistemas de IoT que exigem respostas imediatas.
- **A) Errado**: Ingestão em lote é lenta e processa dados em intervalos, não adequado para necessidades em tempo real.
- **C) Errado**: Data Lakes são usados para armazenar grandes volumes de dados brutos, não para processamento imediato.
- **D) Errado**: MapReduce é usado para processamento distribuído de dados em lote, não em tempo real.

Questão 14

Uma empresa está desenvolvendo uma aplicação que requer consultas complexas sobre grandes volumes de dados e necessita de forte consistência transacional. Qual SGBD seria a melhor escolha?

- A) MongoDB
- B) MySQL
- C) Cassandra
- D) Redis

▶ **Resposta: B) MySQL**

Explicação:

- **B) Correto**: MySQL é um SGBD relacional que suporta transações ACID, garantindo forte consistência e é otimizado para consultas complexas.
- **A) Errado**: MongoDB é NoSQL e não fornece consistência transacional como o MySQL.
- **C) Errado**: Cassandra prioriza disponibilidade e escalabilidade, não consistência forte.
- **D) Errado**: Redis é um banco de dados chave-valor usado principalmente para caching e não oferece suporte robusto a transações complexas.

Questão 15

Para um sistema de análise de sentimentos em redes sociais, onde os dados precisam ser processados rapidamente para fornecer insights quase em tempo real, qual ferramenta é mais adequada?

- A) Apache Hadoop
- B) Apache Hive
- C) Apache Spark
- D) SQL Server

▶ **Resposta: C) Apache Spark**

Explicação:

- **C) Correto**: Apache Spark é ideal para processamento em tempo real e é amplamente usado em análises rápidas como a de sentimentos.

- **A) Errado**: Hadoop é mais lento devido ao uso do MapReduce e é mais adequado para processamento em lote.
- **B) Errado**: Hive é usado para consultas em Hadoop e não é otimizado para tempo real.
- **D) Errado**: SQL Server é bom para transações estruturadas, mas não é otimizado para análises rápidas em grandes volumes de dados.

Questão 16

Qual ferramenta é mais indicada para gerenciar um cluster de servidores de armazenamento de dados, permitindo alta disponibilidade e tolerância a falhas?

- A) MySQL
- B) HDFS
- C) PostgreSQL
- D) MongoDB

▶ **Resposta: B) HDFS**

Explicação:

- **B) Correto**: HDFS (Hadoop Distributed File System) é projetado para armazenar grandes volumes de dados com alta disponibilidade e tolerância a falhas através de replicação de dados entre nós do cluster.
- **A) Errado**: MySQL pode ser configurado para alta disponibilidade, mas não é um sistema de arquivos distribuído.
- **C) Errado**: PostgreSQL também pode ser configurado para alta disponibilidade, mas não é especializado em gerenciar clusters distribuídos.
- **D) Errado**: MongoDB é uma base de dados NoSQL que oferece alta disponibilidade, mas o foco principal de HDFS é o armazenamento distribuído.

Questão 17

Qual modelo de armazenamento é mais eficiente para análises em que a leitura frequente de colunas específicas é necessária, como em relatórios de negócios?

- A) Modelo Relacional
- B) Modelo Colunar
- C) Modelo Orientado a Documentos
- D) Modelo Chave-Valor

▶ **Resposta: B) Modelo Colunar**

Explicação:

- **B) Correto**: O modelo colunar armazena dados por colunas, o que acelera a leitura de colunas específicas, ideal para consultas analíticas frequentes.
- **A) Errado**: O modelo relacional armazena dados em linhas, o que pode ser mais lento para consultas que focam em colunas específicas.

- **C) Errado**: Modelos orientados a documentos não são otimizados para leitura de colunas específicas.
- **D) Errado**: O modelo chave-valor é muito básico para consultas analíticas complexas.

Questão 18

Qual das seguintes opções é um benefício do uso de Data Lakehouse em comparação com Data Lakes e Data Warehouses?

- A) Armazena apenas dados estruturados.
- B) Combina a flexibilidade de Data Lakes com a performance de Data Warehouses.
- C) É exclusivo para armazenamento de dados brutos.
- D) Não oferece suporte para dados não estruturados.

▶ **Resposta: B) Combina a flexibilidade de Data Lakes com a performance de Data Warehouses.**

Explicação:

- **B) Correto**: Data Lakehouses combinam o melhor dos Data Lakes (flexibilidade para dados brutos) e dos Data Warehouses (performance para consultas).
- **A) Errado**: Data Lakehouses suportam tanto dados estruturados quanto não estruturados.
- **C) Errado**: Eles oferecem mais que armazenamento de dados brutos, permitindo também análises rápidas.
- **D) Errado**: Suportam todos os tipos de dados, inclusive não estruturados.

Questão 19

Em uma aplicação que requer alto desempenho na leitura de grandes volumes de dados e suporta operações paralelas, qual modelo é mais adequado?

- A) Modelo Relacional
- B) Modelo Colunar
- C) Modelo Orientado a Grafos
- D) Modelo Orientado a Documentos

▶ **Resposta: B) Modelo Colunar**

Explicação:

- **B) Correto**: Modelos colunares são otimizados para leituras rápidas de grandes volumes de dados, sendo ideais para análises paralelas.
- **A) Errado**: Modelos relacionais não são tão rápidos quanto os colunares em leituras de dados massivos.
- **C) Errado**: Modelos de grafos são melhores para explorar relacionamentos complexos.
- **D) Errado**: Modelos orientados a documentos são flexíveis, mas não têm o desempenho otimizado para consultas massivas como os colunares.

Questão 20

Qual é a principal razão para escolher um Data Mart em vez de um Data Warehouse para uma equipe de vendas?

- A) Maior capacidade de armazenamento.
- B) Focado nas necessidades específicas de um departamento.
- C) Melhor desempenho para análises de big data.
- D) Suporte exclusivo a dados não estruturados.

▶ **Resposta: B) Focado nas necessidades específicas de um departamento.**

Explicação:

- **B) Correto**: Data Marts são projetados para atender necessidades específicas de um departamento, como vendas, oferecendo dados filtrados e relevantes.
- **A) Errado**: Data Warehouses têm maior capacidade de armazenamento.
- **C) Errado**: Data Warehouses são mais adequados para análises complexas de big data.
- **D) Errado**: Ambos podem suportar dados semiestruturados, mas a característica principal do Data Mart é seu foco específico.

Questoes discursivas

Questão 1

Explique o conceito de Big Data e discuta como os "5 Vs" (Volume, Velocidade, Variedade, Veracidade e Valor) caracterizam este fenômeno.

▶ **Resposta**

Big Data refere-se ao conjunto de grandes volumes de dados que são gerados, armazenados e analisados com alta velocidade e em formatos variados. Os "5 Vs" caracterizam o Big Data da seguinte forma:

- **Volume**: Refere-se à quantidade massiva de dados gerados a partir de múltiplas fontes, como redes sociais, sensores IoT e transações financeiras.
- **Velocidade**: Diz respeito à rapidez com que os dados são gerados e precisam ser processados para fornecer informações em tempo real.
- **Variedade**: Representa a diversidade de formatos de dados, incluindo dados estruturados, semiestruturados (JSON, XML) e não estruturados (vídeos, imagens).
- **Veracidade**: Refere-se à qualidade e confiabilidade dos dados, um desafio devido à variabilidade nas fontes.
- **Valor**: Destaca a importância de extrair insights úteis e acionáveis dos dados para apoiar a tomada de decisões.

Questão 2

Descreva as diferenças entre um Data Warehouse e um Data Lake, destacando os casos de uso para cada um.

▶ **Resposta**

Um **Data Warehouse** é um repositório de dados estruturados otimizado para consultas complexas e relatórios. Ele organiza os dados em tabelas e é ideal para análises de dados históricos e relatórios financeiros.

Um **Data Lake**, por outro lado, armazena dados em sua forma bruta, sem a necessidade de estruturação prévia. Ele é adequado para armazenar grandes volumes de dados de várias fontes em diferentes formatos (estruturados, semiestruturados e não estruturados) e é amplamente usado em análises exploratórias e aprendizado de máquina.

- **Casos de uso Data Warehouse**: Relatórios financeiros, análises de vendas e indicadores de desempenho.
- **Casos de uso Data Lake**: Análise de dados de sensores, redes sociais e experimentações com aprendizado de máquina.

Questão 3

Explique o que é um modelo de dados orientado a grafos e forneça um exemplo de aplicação prática onde esse modelo é ideal.

▶ **Resposta**

O modelo de dados orientado a grafos armazena informações como nós (entidades), arestas (relações) e propriedades (atributos de nós e arestas), sendo ideal para representar e consultar conexões complexas entre entidades. Esse modelo é amplamente utilizado para explorar redes sociais, onde se busca identificar conexões entre usuários, como amigos em comum, ou em sistemas de recomendação que identificam itens semelhantes com base nas interações de usuários.

- **Exemplo de aplicação**: Redes sociais como Facebook, onde o modelo de grafos é usado para explorar as conexões entre usuários e para sugerir novas amizades com base nas interações já existentes.

Questão 4

O que é ingestão de dados em lote (batch) e em tempo real (stream)? Compare os dois métodos destacando vantagens e desvantagens.

▶ **Resposta**

Ingestão em lote (batch) processa grandes volumes de dados em intervalos regulares, permitindo que dados sejam coletados, agrupados e processados de uma só vez. Vantagens incluem eficiência em grandes volumes e uso de recursos controlado; desvantagens são o atraso no processamento e a falta de suporte para ações em tempo real.

Ingestão em tempo real (stream) processa dados continuamente assim que são gerados, permitindo decisões imediatas e ações rápidas. Suas vantagens incluem baixa latência e suporte a ações em tempo real; desvantagens envolvem maior complexidade de implementação e maior uso contínuo de recursos.

- **Batch** é ideal para relatórios periódicos.
- **Stream** é essencial para monitoramento em tempo real, como detecção de fraudes.

Questão 5

Descreva a função do HDFS (Hadoop Distributed File System) no contexto do Big Data e como ele garante a tolerância a falhas.

▶ **Resposta**

O HDFS é o sistema de arquivos distribuído do Hadoop, projetado para armazenar grandes volumes de dados em clusters de servidores. Ele divide os dados em blocos e os distribui pelos nós do cluster, replicando-os para garantir a disponibilidade e tolerância a falhas. Se um nó falhar, os dados ainda podem ser acessados a partir das réplicas em outros nós, garantindo a continuidade do processamento sem perda de informações.

Questão 6

Explique o que são metadados em bases de dados e qual é sua importância na gestão de dados.

▶ **Resposta**

Metadados são informações que descrevem as características de outros dados, como tipo, formato, estrutura e restrições. Em bases de dados, metadados definem a estrutura das tabelas, tipos de colunas, chaves primárias e estrangeiras, e regras de integridade. Eles são essenciais para garantir a organização, compreensão, e consistência dos dados, facilitando a gestão e o uso eficiente da base de dados.

Questão 7

Discuta as principais características de um SGBD NoSQL e seus casos de uso mais comuns.

▶ **Resposta**

SGBDs NoSQL são sistemas de gerenciamento de bases de dados projetados para lidar com grandes volumes de dados não estruturados ou semiestruturados. Suas principais características incluem flexibilidade de esquema, alta escalabilidade horizontal, e desempenho otimizado para grandes volumes de leitura e escrita.

- **Casos de uso comuns**: Aplicações de redes sociais, armazenando posts e interações em formatos flexíveis, e sistemas de e-commerce, onde a rápida adaptação do modelo de dados é necessária para novos produtos e categorias.

Questão 8

Defina o conceito de Data Lakehouse e explique como ele combina características de Data Lakes e Data Warehouses.

▶ **Resposta**

Data Lakehouse é uma arquitetura de armazenamento que combina a flexibilidade de Data Lakes com a performance e governança de Data Warehouses. Ele permite armazenar dados brutos em formatos

variados, ao mesmo tempo que oferece suporte a consultas analíticas rápidas sobre esses dados, com funcionalidades de governança e otimização que são típicas de Data Warehouses.

Questão 9

Explique como o Apache Spark se diferencia do Hadoop MapReduce em termos de processamento de dados.

▶ **Resposta**

O Apache Spark diferencia-se do Hadoop MapReduce principalmente pela sua capacidade de processar dados em memória, o que reduz significativamente o tempo de execução de tarefas, especialmente para análises repetitivas ou interativas. Enquanto o MapReduce utiliza disco para cada etapa do processamento, o Spark mantém os dados em memória, acelerando o processamento e permitindo o uso para análise em tempo real e aprendizado de máquina.

Questão 10

Discuta os principais usos e vantagens do modelo colunar de armazenamento de dados em comparação com o modelo relacional tradicional.

▶ **Resposta**

O modelo colunar armazena dados por colunas em vez de linhas, otimizando o armazenamento e a leitura de colunas específicas, o que é vantajoso para consultas analíticas. Comparado ao modelo relacional tradicional, que armazena dados em linhas, o modelo colunar reduz o volume de dados lidos para consultas que envolvem apenas algumas colunas, aumentando significativamente o desempenho de análises e relatórios complexos, especialmente em grandes volumes de dados.