ibmec.br

Inovação e as Bases de IA, Data Science e Big Data

Aula 3

Professor: Marcius Linhares

Ibmec

Conteúdo

Nº AULA	DATA	CONTEÚDO
1	27/08/2024 (terça-feira)	Introdução à Inovação, Business Intelligence e Tecnologia
2	28/08/2024 (quarta-feira)	Fundamentos de Inteligência Artificial e Soluções Emergentes
3	03/09/2024 (terça-feira)	Fundamentos de Data Science
4	04/09/2024 (quarta-feira)	Fundamentos de Big Data
5	10/09/2024 (terça-feira)	Integração de IA, Data Science e Big Data na Inovação
6	11/09/2024 (quarta-feira)	Tendências, Ética e o Futuro da IA e Big Data / Inicio Trabalho em Grupo

Fundamentos de Ciência de Dados

A **Ciência de Dados** é um campo interdisciplinar que combina conhecimentos de estatística, matemática, ciência da computação e habilidades de domínio específico para extrair insights e conhecimento a partir de dados. O principal objetivo da ciência de dados é transformar grandes volumes de dados em informações úteis e acionáveis para apoiar a tomada de decisões.

1. Importância da Linguagem SQL para o Cientista de Dados

SQL (Structured Query Language) é uma linguagem essencial para cientistas de dados, especialmente porque a grande maioria dos dados nas empresas estão em bancos de dados relacionais. SQL permite que cientistas de dados acessem, manipulem e analisem grandes volumes de dados de forma eficiente, sendo uma ferramenta indispensável no dia a dia.

2. Principais Comandos SQL

SELECT: Usado para selecionar dados de uma ou mais tabelas.

Exemplo:

Selecionar todos os registros de uma tabela de produtos

SELECT * FROM produtos;
Selecionar apenas o nome e o preço dos produtos

SELECT nome, preco FROM produtos;

Selecionar todos os produtos com preço maior que 100

SELECT nome, preco FROM produtos WHERE preco > 100;

INSERT: Usado para inserir novos registros em uma tabela.

Exemplo: **Inserir um novo produto na tabela**

```
INSERT INTO produtos (nome, preco, descricao) VALUES
('Notebook', 3000,00, 'Notebook de última geração com 16GB
de RAM');
```

UPDATE: Usado para atualizar registros existentes em uma tabela.

Exemplo: **Atualizar o preço de um produto**

```
UPDATE produtos SET preco = 3200, 00 WHERE nome =
'Notebook';
```

DELETE: Usado para deletar registros de uma tabela.

Exemplo: **Deletar um produto da tabela**

```
DELETE FROM produtos WHERE nome = 'Notebook';
```

3.Comandos de JOIN em SQL

JOINs são usados para combinar dados de duas ou mais tabelas com base em uma condição relacionada. Eles são extremamente importantes para criar relatórios complexos e analisar dados de diferentes fontes.

INNER JOIN: Retorna os registros que têm correspondência em ambas as tabelas.

Exemplo:

Selecionar pedidos e os nomes dos clientes que os fizeram

```
SELECT pedidos.id_pedido, clientes.nome

FROM pedidos INNER JOIN clientes ON pedidos.id_cliente =
clientes.id_cliente;
```

LEFT JOIN (ou LEFT OUTER JOIN): Retorna todos os registros da tabela à esquerda e os registros correspondentes da tabela à direita. Se não houver correspondência, NULL é retornado para as colunas da tabela à direita.

Exemplo: **Selecionar todos os clientes e os pedidos correspondentes (se houver)**

```
SELECT clientes.nome, pedidos.id_pedido
FROM clientes LEFT JOIN pedidos ON clientes.id_cliente =
pedidos.id_cliente;
```

RIGHT JOIN (ou RIGHT OUTER JOIN): Retorna todos os registros da tabela à direita e os registros correspondentes da tabela à esquerda. Se não houver correspondência, NULL é retornado para as colunas da tabela à esquerda.

Exemplo: **Selecionar todos os pedidos e os nomes dos clientes correspondentes (se houver)**

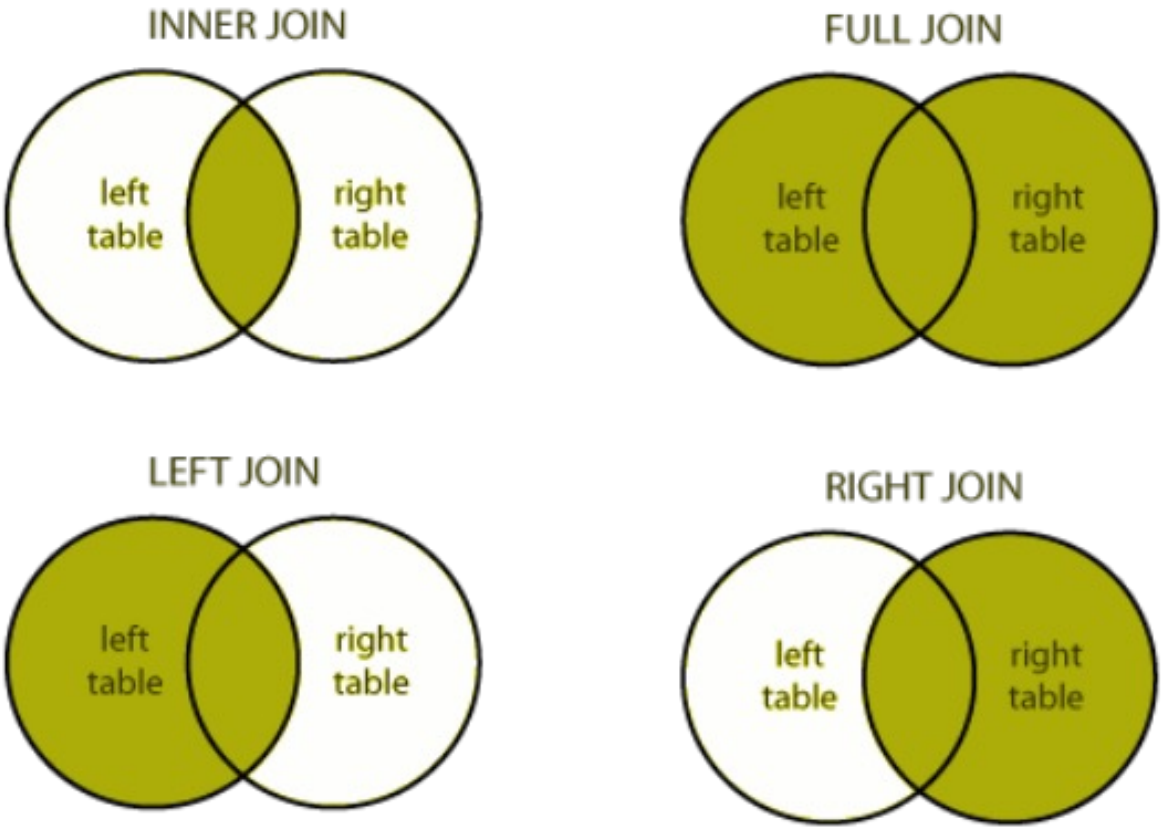
```
SELECT pedidos.id_pedido, clientes.nome
FROM pedidos RIGHT JOIN clientes ON pedidos.id_cliente
= clientes.id_cliente;
```

FULL OUTER JOIN: Retorna todos os registros quando há uma correspondência em uma das tabelas. Se não houver correspondência, NULL é retornado para a tabela que não tiver correspondência.

Exemplo: **Selecionar todos os clientes e todos os pedidos, independentemente de haver correspondência**

```
SELECT clientes.nome, pedidos.id_pedido
FROM clientes FULL OUTER JOIN pedidos ON clientes.id_cliente =
pedidos.id_cliente;
```

Aula 3



O que é Agregação no SQL?

Agregação no SQL refere-se ao processo de combinar múltiplas linhas de dados em uma única linha de resumo ou total. Isso é feito usando funções de agregação, que executam cálculos em um conjunto de valores e retornam um único valor. As funções de agregação são extremamente úteis para resumir e analisar grandes volumes de dados.

Principais Funções de Agregação no SQL:

SUM: Calcula a soma de um conjunto de valores.

AVG: Calcula a média de um conjunto de valores.

COUNT: Conta o número de linhas que correspondem a um critério.

MAX: Retorna o maior valor em um conjunto de valores.

MIN: Retorna o menor valor em um conjunto de valores.

Exemplos Práticos de Agregação no SQL

SUM: Soma de Valores

Exemplo: Suponha que você tenha uma tabela vendas e queira calcular o total de vendas de um determinado produto.

```
SELECT SUM(valor) AS total_vendas
FROM vendas
WHERE produto = 'Notebook';
```

AVG: Média de Valores

Exemplo: Calcule a média de vendas diárias em uma tabela de vendas.

Calcular a média diária de vendas

```
SELECT AVG(valor) AS media_vendas_diarias FROM vendas
WHERE data_venda BETWEEN '2024-01-01' AND '2024-01-31';
```

COUNT: Contagem de Linhas

Exemplo: Conte quantos pedidos foram feitos por um determinado cliente.

Contar o número de pedidos feitos pelo cliente 'João'

```
SELECT COUNT(*) AS total_pedidos
FROM pedidos WHERE cliente = 'João';
```

Uso de Agregação com GROUP BY

A agregação se torna ainda mais poderosa quando combinada com a cláusula GROUP BY, que permite agrupar os dados antes de aplicar a função de agregação. Isso é particularmente útil quando você deseja calcular valores agregados para diferentes grupos de dados, como total de vendas por produto ou média de salário por departamento.

Exemplo de GROUP BY com SUM

Exemplo: Calcule o total de vendas por produto em uma tabela de vendas.

Calcular o total de vendas para cada produto

```
SELECT produto, SUM(valor) AS total_vendas
FROM vendas
GROUP BY produto;
```

Uma **biblioteca Python** é uma coleção de módulos e pacotes que contêm funções, classes, e variáveis pré-escritas, projetadas para realizar tarefas específicas, que os desenvolvedores podem reutilizar em seus próprios programas. Essas bibliotecas permitem que os desenvolvedores evitem escrever código do zero para tarefas comuns, acelerando o processo de desenvolvimento e garantindo a consistência e a eficiência do código.

Aula 3

Principais Características de uma Biblioteca Python:

1. Reutilização de Código:

1. Bibliotecas contêm código que foi escrito, testado e otimizado por outros desenvolvedores. Isso permite que você reutilize esse código em seus próprios projetos, economizando tempo e esforço.

2. Modularidade:

1. As bibliotecas são compostas de módulos, que são arquivos Python contendo funções e classes relacionadas. Isso significa que você pode importar apenas as partes da biblioteca que precisa, mantendo seu código organizado e eficiente.

Ampla Gama de Funcionalidades:

- Existem bibliotecas Python para praticamente qualquer tarefa, desde manipulação de dados e visualização, até aprendizado de máquina, processamento de linguagem natural, e automação de tarefas.

- **Facilidade de Uso:**

- Bibliotecas Python geralmente vêm com documentação completa e exemplos que facilitam seu uso, mesmo para iniciantes.

1 – Pandas

Por que usar? Manipulação e análise de dados tabulares, como folhas de cálculo ou tabelas SQL, é muito comum em ciência de dados. Pandas facilita a manipulação desses dados de forma eficiente e flexível.

Link oficial: https://pandas.pydata.org/docs/index.html

2 – NumPy

- **Por que usar?** Quando se lida com grandes volumes de dados numéricos, como vetores e matrizes, a eficiência e a velocidade são cruciais. NumPy fornece suporte para operações matemáticas rápidas em arrays multidimensionais.

Link oficial: https://numpy.org/

3 – Matplotlib

Por que usar? A visualização de dados é essencial para entender tendências, padrões e outliers. Matplotlib permite criar uma ampla variedade de gráficos, que são fundamentais para a análise de dados.

Link oficial: https://matplotlib.org/

4 - Seaborn

Por que usar? Para criar gráficos estatísticos avançados e visualmente atraentes, Seaborn oferece uma interface mais simplificada e estilizada do que Matplotlib, além de ser excelente para explorar relações entre variáveis.

Link oficial: https://seaborn.pydata.org/

5 - Scikit-learn

Por que usar? Scikit-learn é a biblioteca de escolha para aprendizado de máquina em Python, oferecendo uma implementação fácil e eficiente de uma ampla gama de algoritmos, desde regressão linear até clusterização.

Link oficial: https://scikit-learn.org

6 - TensorFlow

Por que usar? TensorFlow é amplamente utilizado para construir e treinar modelos complexos de deep learning, como redes neurais profundas, que são essenciais para tarefas como reconhecimento de voz, visão computacional e processamento de linguagem natural.

Link oficial: https://www.tensorflow.org/?hl=pt-br

7 - Keras

Por que usar? Keras é uma API de alto nível que facilita a prototipagem e o desenvolvimento de modelos de deep learning. É ideal para iniciantes e para quem precisa construir rapidamente modelos complexos em TensorFlow.

Link oficial: https://keras.io/

8 – PyTorch

Por que usar? PyTorch oferece flexibilidade e simplicidade, especialmente útil para pesquisa em deep learning. Ele é popular por sua abordagem de computação dinâmica, que facilita a experimentação e a depuração.

Link oficial: https://pytorch.org/

9 - Statsmodels

Por que usar? Statsmodels é essencial para análises estatísticas rigorosas. Ela permite a construção de modelos estatísticos, realização de testes de hipótese e análise de séries temporais com uma base matemática sólida.

Link oficial: https://www.statsmodels.org/

10 – SciPy

Por que usar? SciPy complementa o NumPy ao fornecer algoritmos avançados para otimização, integração, interpolação, álgebra linear, e outras operações matemáticas complexas.

Link oficial: https://scipy.org/

11 – NLTK (Natural Language Toolkit)

Por que usar? NLTK é ideal para tarefas de processamento de linguagem natural (NLP), como análise de texto, tokenização, e análise de sentimentos, oferecendo uma ampla gama de ferramentas linguísticas.

Link oficial: https://www.nltk.org/

12 – BeautifulSoup

Por que usar? BeautifulSoup é uma ferramenta essencial para web scraping, permitindo a extração de dados estruturados de páginas HTML e XML, facilitando a coleta de informações da web.

Link oficial: https://beautiful-soup-4.readthedocs.io/en/latest/

Introdução ao CRISP-DM (Cross-Industry Standard Process for Data Mining):

Conteúdo:

CRISP-DM é uma metodologia amplamente utilizada para o desenvolvimento de projetos de Data Science. Consiste em seis fases principais que guiam o processo de extração de conhecimento a partir de dados.

Aula 3

Entendimento do Negócio: Compreender os objetivos do projeto e os requisitos do negócio.

Entendimento dos Dados: Coleta e compreensão dos dados disponíveis para alcançar os objetivos.

Preparação dos Dados: Limpeza, transformação e organização dos dados para análise.

Modelagem: Aplicação de técnicas de modelagem para construir modelos preditivos ou descritivos.

Avaliação: Avaliação dos modelos e comparação com os objetivos do negócio.

Implantação: Implementação dos modelos em ambiente de produção e monitoramento dos resultados.

Exemplo: Aplicando CRISP-DM para analisar os gastos parlamentares e identificar padrões de despesas, desde o entendimento dos requisitos do projeto até a implantação de um dashboard de monitoramento.

https://dadosabertos.camara.leg.br/swagger/api.html?tab=staticfile

Definição de Problemas (Entendimento do Negócio):

Conteúdo: Identificação e formulação do problema a ser resolvido com Data Science. Com CRISP-DM, esta fase envolve discussões com stakeholders para definir claramente os objetivos.

Exemplo: Determinar quais parlamentares têm o maior gasto médio mensal e identificar possíveis irregularidades.

Coleta e Entendimento dos Dados:

Conteúdo: Coleta de dados a partir de fontes públicas e compreensão dos mesmos. No contexto do CRISP-DM, essa fase envolve a análise inicial dos dados para verificar sua qualidade e adequação.

Exemplo: Download e exploração inicial dos dados de gastos parlamentares da Câmara dos Deputados.

Preparação dos Dados:

Conteúdo: Limpeza e transformação dos dados para análise, alinhando-se com a fase de Preparação dos Dados no CRISP-DM.

Exemplo: Conversão das colunas de datas para o formato datetime, preenchimento de valores ausentes e normalização de valores monetários.

Modelagem:

Conteúdo: Aplicação de técnicas de modelagem, como a criação de modelos preditivos ou segmentação de dados, de acordo com a fase de Modelagem no CRISP-DM.

Exemplo: Uso de técnicas de clusterização para agrupar parlamentares com perfis de gastos semelhantes.

Avaliação:

Conteúdo: Avaliação dos modelos e resultados da análise utilizando métricas apropriadas, conforme a fase de Avaliação do CRISP-DM.

Exemplo: Avaliar a precisão de um modelo preditivo que estima os gastos de um parlamentar com base em seu histórico.

Implantação:

Conteúdo: Publicação dos resultados ou implementação dos modelos em ambientes de produção, como descrito na fase de Implantação do CRISP-DM.

Exemplo: Desenvolvimento de um dashboard onde se pode monitorar os gastos parlamentares em tempo real.

Aula 3

ibmec.br

Sites Importantes para um Cientista de Dados

Aula 3

Kaggle

Descrição: Kaggle é uma plataforma de ciência de dados que oferece competições, datasets, tutoriais e notebooks de código compartilhados pela comunidade. É uma excelente plataforma para praticar habilidades de ciência de dados e aprendizado de máquina, bem como para aprender com outros profissionais.

Link: www.kaggle.com

Towards Data Science

Descrição: Uma publicação da plataforma Medium, Towards Data Science apresenta artigos, tutoriais e casos de uso escritos por profissionais da área. É uma excelente fonte de aprendizado contínuo e de exploração de novas ideias em ciência de dados.

Link: towardsdatascience.com

Aula 3

GitHub

Descrição: GitHub é uma plataforma de hospedagem de código que permite colaboração entre desenvolvedores. Cientistas de dados podem encontrar e compartilhar código, explorar repositórios de projetos de ciência de dados e contribuir para projetos de código aberto.

Link: www.github.com

Stack Overflow

Descrição: Stack Overflow é uma comunidade de perguntas e respostas para programadores e desenvolvedores. Cientistas de dados podem encontrar respostas para problemas específicos de programação e ciência de dados ou ajudar outros compartilhando seu conhecimento.

Link: www.stackoverflow.com

Aula 3

Coursera

Descrição: Coursera oferece cursos online de instituições renomadas sobre ciência de dados, aprendizado de máquina, estatística, e mais. Os cursos podem ser realizados no seu próprio ritmo, e muitos oferecem certificações.

Link: www.coursera.org/browse/data-science

DataCamp

Descrição: DataCamp é uma plataforma de aprendizado online que oferece cursos interativos em ciência de dados e programação. Focado em Python, R, SQL, e outras ferramentas de ciência de dados, DataCamp é ideal para quem deseja praticar e aprender através de exercícios práticos.

Link: www.datacamp.com

Aula 3

Analytics Vidhya

Descrição: Analytics Vidhya é uma comunidade que oferece tutoriais, blogs, competições e cursos relacionados a ciência de dados e aprendizado de máquina. É uma boa fonte de recursos para iniciantes e profissionais.

Link: www.analyticsvidhya.com

Data Science Central

Descrição: Data Science Central é uma comunidade online para profissionais de dados que fornece artigos, webinars, discussões, e notícias sobre ciência de dados, big data, e inteligência artificial.

Link: www.datasciencecentral.com

Github das Aulas

https://github.com/marciuslinhares/IBMEC-INOVACAO-E-AS-BASES-DE-IA-DATA-SCIENCE-E-BIG-DATA

IBMEC.BR

/IBMEC

IBMEC

@IBMEC_OFICIAL

@IBMEC

Ibmec