Questões de Python: Do Básico ao Avançado

50 questões de múltipla escolha, divididas em 10 níveis de dificuldade crescente, para testar seus conhecimentos em Python.

**Nível 1: Fundamentos (1-5)**

**1. Qual é a forma correta de exibir a frase "Olá, mundo!" no console em Python?**  
a) print("Olá, mundo!")  
b) console.log("Olá, mundo!")  
c) echo("Olá, mundo!")  
d) System.out.println("Olá, mundo!")  
e) printf("Olá, mundo!")

**2. Qual é o tipo de dado da variável idade = 25?**  
a) String  
b) Float  
c) Integer  
d) Boolean  
e) List

**3. Qual operador realiza a exponenciação em Python?**  
a) ^  
b) \*\*  
c) //  
d) %  
e) \*

**4. O que o código print(10 / 3) irá exibir?**  
a) 3  
b) 3.33  
c) 3.3333333333333335  
d) 3.0  
e) Erro

**5. Qual a função utilizada para ler dados do usuário no console?**  
a) read()  
b) input()  
c) get()  
d) scan()  
e) readline()

**Nível 2: Estruturas de Controle (6-10)**

**6. Qual estrutura de controle é utilizada para executar um bloco de código repetidamente enquanto uma condição for verdadeira?**  
a) if  
b) elif  
c) else  
d) for  
e) while

**7. Qual é o valor final de x após a execução do código?**

x = 0

while x < 5:

x += 2

a) 2  
b) 4  
c) 5  
d) 6  
e) 8

**8. O que o código a seguir irá imprimir?**

for i in range(1, 4):

print(i)

a) 0 1 2 3  
b) 1 2 3  
c) 1 2 3 4  
d) 0 1 2  
e) Erro

**9. Qual a estrutura de controle utilizada para interromper a execução de um loop?**  
a) continue  
b) break  
c) pass  
d) stop  
e) exit

**10. Qual a finalidade da cláusula else em um bloco try...except?**  
a) Executar código se ocorrer uma exceção específica.  
b) Executar código se não ocorrer nenhuma exceção.  
c) Executar código após o bloco try ser finalizado, independentemente de exceções.  
d) Definir um bloco de código a ser executado se o bloco except falhar.  
e) Não é possível utilizar else em um bloco try...except.

**Nível 3: Estruturas de Dados (11-15)**

**11. Qual estrutura de dados é uma coleção ordenada e mutável em Python?**  
a) Lista  
b) Tupla  
c) Conjunto  
d) Dicionário  
e) String

**12. Qual método adiciona um elemento ao final de uma lista?**  
a) add()  
b) append()  
c) insert()  
d) extend()  
e) push()

**13. O que diferencia uma tupla de uma lista em Python?**  
a) Tuplas são ordenadas e listas não são.  
b) Tuplas são imutáveis e listas são mutáveis.  
c) Tuplas permitem elementos repetidos e listas não permitem.  
d) Tuplas utilizam colchetes [] e listas utilizam parênteses ().  
e) Não há diferença entre tuplas e listas em Python.

**14. Qual estrutura de dados utiliza pares chave-valor?**  
a) Lista  
b) Tupla  
c) Conjunto  
d) Dicionário  
e) String

**15. O que o código a seguir irá imprimir?**

conjunto = {1, 2, 2, 3}

print(conjunto)

a) {1, 2, 2, 3}  
b) {1, 2, 3}  
c) [1, 2, 2, 3]  
d) [1, 2, 3]  
e) Erro

**Nível 4: Funções (16-20)**

**16. Qual a palavra-chave utilizada para definir uma função em Python?**  
a) function  
b) def  
c) define  
d) fun  
e) func

**17. O que o código a seguir irá imprimir?**

def saudacao():

return "Olá!"

print(saudacao())

a) Olá!  
b) saudacao()  
c) None  
d) Erro  
e) Nada será impresso.

**18. Qual a finalidade da instrução return em uma função?**  
a) Exibir um valor no console.  
b) Finalizar a execução da função.  
c) Retornar um valor da função para o código que a chamou.  
d) Atribuir um valor a uma variável global.  
e) return não é uma instrução válida em Python.

**19. O que são argumentos em uma função?**  
a) Valores retornados pela função.  
b) Variáveis locais da função.  
c) Valores passados para a função quando ela é chamada.  
d) Instruções executadas dentro da função.  
e) Comentários que documentam o funcionamento da função.

**20. Qual a diferença entre \*args e \*\*kwargs em uma função?**  
a) \*args é usado para argumentos nomeados e \*\*kwargs para argumentos posicionais.  
b) \*args é usado para um número variável de argumentos posicionais e \*\*kwargs para um número variável de argumentos nomeados.  
c) \*args é usado para listas e \*\*kwargs para dicionários.  
d) \*args e \*\*kwargs têm a mesma finalidade, sendo apenas convenções de nomenclatura.  
e) Não é possível utilizar \*args e \*\*kwargs na mesma função.

**Nível 5: Escopo de Variáveis (21-25)**

**21. O que é o escopo de uma variável em Python?**  
a) O tipo de dado da variável.  
b) O valor atribuído à variável.  
c) A região do código onde a variável é válida e pode ser acessada.  
d) A visibilidade da variável em relação a outros arquivos do projeto.  
e) A capacidade da variável de armazenar diferentes tipos de dados durante a execução.

**22. O que o código a seguir irá imprimir?**

x = 10

def minha\_funcao():

x = 5

print(x)

minha\_funcao()

print(x)

a) 5 5  
b) 10 10  
c) 5 10  
d) 10 5  
e) Erro

**23. Como acessar uma variável global dentro de uma função?**  
a) Utilizando a palavra-chave global antes do nome da variável.  
b) Passando a variável como argumento para a função.  
c) Utilizando o operador de acesso . antes do nome da variável.  
d) Não é possível acessar variáveis globais dentro de funções.  
e) Variáveis globais são acessadas automaticamente dentro de funções.

**24. O que o código a seguir irá imprimir?**

def minha\_funcao():

global x

x = 10

minha\_funcao()

print(x)

a) 5  
b) 10  
c) None  
d) Erro  
e) Nada será impresso.

**25. Qual a melhor prática em relação ao uso de variáveis globais em funções?**  
a) Utilizar variáveis globais sempre que possível para facilitar o acesso aos dados.  
b) Evitar o uso de variáveis globais sempre que possível, pois elas podem tornar o código mais difícil de entender e manter.  
c) Utilizar variáveis globais apenas para constantes.  
d) Não há problema em utilizar variáveis globais em funções, desde que sejam declaradas com a palavra-chave global.  
e) O uso de variáveis globais é irrelevante para a qualidade do código.

**Nível 6: Manipulação de Strings (26-30)**

**26. Qual o método utilizado para concatenar (juntar) duas strings?**  
a) join()  
b) concat()  
c) +  
d) append()  
e) merge()

**27. O que o código a seguir irá imprimir?**

frase = "Python é legal!"

print(frase[7:10])

a) Python  
b) é  
c) legal  
d) legal!  
e) thon

**28. Qual método converte uma string para letras minúsculas?**  
a) lower()  
b) lowercase()  
c) toLower()  
d) str.lower()  
e) str\_lower()

**29. O que o código a seguir irá imprimir?**

frase = "Olá, mundo!"

print(len(frase))

a) 10  
\* b) 11  
c) 12  
d) 13  
e) Erro

**30. Qual método verifica se uma string começa com um determinado prefixo?**  
a) startswith()  
b) beginswith()  
c) hasPrefix()  
d) isPrefix()  
e) checkPrefix()

**Nível 7: Trabalhando com Arquivos (31-35)**

**31. Qual a função utilizada para abrir um arquivo em Python?**  
a) file()  
b) open()  
c) read()  
d) write()  
e) load()

**32. Qual o modo de abertura de arquivo utilizado para escrever dados em um novo arquivo?**  
a) 'r'  
b) 'w'  
c) 'a'  
d) 'x'  
e) '+'

**33. O que acontece se você tentar abrir um arquivo que não existe no modo de leitura ('r')?**  
a) O arquivo é criado automaticamente.  
b) Uma exceção FileNotFoundError é lançada.  
c) Um erro de sintaxe é retornado.  
d) O programa entra em loop infinito.  
e) Nada acontece.

**34. Qual método é utilizado para ler o conteúdo completo de um arquivo?**  
a) read()  
b) readline()  
c) readlines()  
d) load()  
e) get()

**35. Qual a boa prática para garantir que um arquivo seja fechado após o uso, mesmo se ocorrerem exceções durante a manipulação do arquivo?**  
a) Utilizar um bloco try...except para capturar exceções e fechar o arquivo no bloco finally.  
b) Não é necessário fechar o arquivo explicitamente, pois o Python faz isso automaticamente.  
c) Fechar o arquivo utilizando close() imediatamente após a última operação de leitura ou escrita.  
d) Utilizar a instrução with open(...) as arquivo:, que fecha o arquivo automaticamente ao sair do bloco with.  
e) Não é possível garantir que um arquivo seja fechado automaticamente em caso de exceções.

**Nível 8: Programação Orientada a Objetos (36-40)**

**36. O que define uma classe em Python?**  
a) Um tipo de dado primitivo.  
b) Um bloco de código reutilizável.  
c) Um conjunto de variáveis globais.  
d) Um modelo para criar objetos, definindo seus atributos e métodos.  
e) Uma estrutura de dados para armazenar pares chave-valor.

**37. O que é um objeto em Python?**  
a) Uma instância de uma classe.  
b) Uma variável global.  
c) Um método estático.  
d) Uma função aninhada.  
e) Um módulo importado.

**38. Qual o método especial utilizado para inicializar os atributos de um objeto durante sua criação?**  
a) \_\_init\_\_()  
b) \_\_new\_\_()  
c) \_\_create\_\_()  
d) \_\_start\_\_()  
e) \_\_construct\_\_()

**39. O que é herança em programação orientada a objetos?**  
a) A capacidade de um objeto acessar atributos e métodos de outros objetos.  
b) A capacidade de uma classe herdar atributos e métodos de outra classe.  
c) A capacidade de um método chamar a si mesmo.  
d) A capacidade de um programa ser executado em diferentes sistemas operacionais.  
e) A capacidade de um código ser facilmente compreendido por outros programadores.

**40. O que é polimorfismo em programação orientada a objetos?**  
a) A capacidade de um objeto ter múltiplos tipos de dados.  
b) A capacidade de uma classe ter múltiplos construtores.  
c) A capacidade de um método ter diferentes implementações em classes diferentes.  
d) A capacidade de um programa ter diferentes comportamentos dependendo da entrada do usuário.  
e) A capacidade de um código ser modularizado em funções e classes.

**Nível 9: Bibliotecas Essenciais (41-45)**

**41. Qual biblioteca é utilizada para realizar cálculos matemáticos mais complexos em Python?**  
a) math  
b) numbers  
c) statistics  
d) numpy  
e) scipy

**42. Qual biblioteca é utilizada para manipular e analisar dados em formato tabular (como dados de um arquivo CSV)?**  
a) openpyxl  
b) csv  
c) pandas  
d) numpy  
e) matplotlib

**43. Qual biblioteca é utilizada para criar gráficos e visualizações de dados em Python?**  
a) pillow  
b) opencv  
c) seaborn  
d) matplotlib  
e) plotly

**44. Qual biblioteca é utilizada para fazer requisições HTTP e interagir com APIs em Python?**  
a) urllib  
b) requests  
c) socket  
d) http  
e) web

**45. Qual biblioteca é utilizada para trabalhar com expressões regulares em Python?**  
a) regex  
b) regexp  
c) re  
d) pattern  
e) string

**Nível 10: Tópicos Avançados (46-50)**

**46. Qual a finalidade de um decorador em Python?**  
a) Adicionar informações de depuração a uma função.  
b) Modificar o comportamento de uma função sem alterá-la diretamente.  
c) Converter uma função em um método de classe.  
d) Definir uma função aninhada dentro de outra função.  
e) Criar uma nova instância de uma classe.

**47. O que é um gerador em Python?**  
a) Uma função que retorna uma lista de valores.  
b) Uma função que gera uma sequência de valores sob demanda.  
c) Uma classe que implementa um iterador.  
d) Uma estrutura de dados para armazenar dados em ordem cronológica.  
e) Um algoritmo para gerar números aleatórios.

**48. Qual a diferença entre multithreading e multiprocessing em Python?**  
a) Multithreading executa múltiplas threads em paralelo em um único processo, enquanto multiprocessing executa múltiplos processos em paralelo.  
b) Multithreading é utilizado para tarefas intensivas de CPU e multiprocessing para tarefas intensivas de E/S.  
c) Multithreading é mais rápido que multiprocessing.  
d) Multiprocessing é mais fácil de implementar que multithreading.  
e) Não há diferença significativa entre multithreading e multiprocessing.

**49. O que é uma exceção em Python?**  
a) Um erro de sintaxe no código.  
b) Um evento que ocorre durante a execução do programa, interrompendo o fluxo normal.  
c) Um aviso que indica um possível problema no código.  
d) Um comentário que documenta uma parte crítica do código.  
e) Uma diretiva que instrui o interpretador Python a ignorar um erro.

**50. Qual a finalidade de um bloco try...except em Python?**  
a) Definir um bloco de código que será executado apenas se uma condição for verdadeira.  
b) Iterar sobre os elementos de uma sequência.  
c) Capturar e tratar exceções que podem ocorrer durante a execução de um bloco de código.  
d) Definir uma função recursiva.  
e) Criar um novo tipo de dado personalizado.

**Gabarito:**

**Nível 1:**

1. **a) print("Olá, mundo!")** - A função print() é usada para exibir dados no console.
2. **c) Integer** - O valor 25 é um número inteiro, representado pelo tipo int em Python.
3. **b) \*\*** - O operador \*\* é utilizado para calcular a potência de um número.
4. **c) 3.3333333333333335** - A divisão em Python retorna um número de ponto flutuante, mesmo que o resultado seja um número inteiro.
5. **b) input()** - A função input() lê uma linha de entrada do usuário no console.

**Nível 2:**  
6. **e) while** - O loop while executa um bloco de código repetidamente enquanto uma condição for verdadeira.  
7. **d) 6** - O loop adiciona 2 a x até que x seja igual a 6, momento em que a condição do loop se torna falsa.  
8. **b) 1 2 3** - A função range(1, 4) gera uma sequência de números de 1 a 3 (excluindo o limite superior).  
9. **b) break** - A instrução break interrompe a execução do loop atual e passa o controle para a próxima instrução após o loop.  
10. **b) Executar código se não ocorrer nenhuma exceção.** - A cláusula else em um bloco try...except é executada apenas se nenhuma exceção for lançada no bloco try.

**Nível 3:**  
11. **a) Lista** - Listas em Python são coleções ordenadas e mutáveis, o que significa que você pode alterar seus elementos após a criação.  
12. **b) append()** - O método append() adiciona um elemento ao final da lista.  
13. **b) Tuplas são imutáveis e listas são mutáveis.** - Uma vez criada, você não pode modificar os elementos de uma tupla, enquanto as listas podem ser modificadas livremente.  
14. **d) Dicionário** - Dicionários em Python são coleções não ordenadas de pares chave-valor.  
15. **b) {1, 2, 3}** - Conjuntos em Python não permitem elementos duplicados, então o conjunto final conterá apenas os elementos únicos.

**Nível 4:**  
16. **b) def** - A palavra-chave def é usada para definir uma nova função em Python.  
17. **a) Olá!** - A função saudacao() retorna a string "Olá!", que é então impressa pela função print().  
18. **c) Retornar um valor da função para o código que a chamou.** - A instrução return permite que uma função envie um resultado de volta para o código que a chamou.  
19. **c) Valores passados para a função quando ela é chamada.** - Argumentos são como "ingredientes" que você fornece a uma função para que ela possa realizar sua tarefa.  
20. **b) \*args é usado para um número variável de argumentos posicionais e \*\*kwargs para um número variável de argumentos nomeados.** - \*args coleta argumentos posicionais extras em uma tupla, enquanto \*\*kwargs coleta argumentos nomeados extras em um dicionário.

**Nível 5:**  
21. **c) A região do código onde a variável é válida e pode ser acessada.** - O escopo determina onde no código uma variável pode ser acessada. Variáveis ​​definidas dentro de uma função têm escopo local, enquanto variáveis ​​definidas fora de qualquer função têm escopo global.  
22. **c) 5 10** - A função minha\_funcao() imprime o valor local de x (5), enquanto a instrução print(x) fora da função imprime o valor global de x (10).  
23. **a) Utilizando a palavra-chave global antes do nome da variável.** - A instrução global dentro de uma função indica que você deseja acessar e modificar a variável global com o mesmo nome.  
24. **b) 10** - A função minha\_funcao() utiliza a instrução global para modificar a variável global x, definindo seu valor como 10.  
25. **b) Evitar o uso de variáveis globais sempre que possível, pois elas podem tornar o código mais difícil de entender e manter.** - Variáveis ​​globais podem tornar o código mais difícil de testar e depurar, pois seu valor pode ser alterado em diferentes partes do programa.

**Nível 6:**  
26. **c) +** - O operador + pode ser usado para concatenar (juntar) duas strings.  
27. **b) é** - A fatia [7:10] seleciona os caracteres da posição 7 até a posição 9 (excluindo a posição 10).  
28. **a) lower()** - O método lower() retorna uma nova string com todos os caracteres em minúsculas.  
29. **c) 12** - A função len() retorna o número de caracteres em uma string.  
30. **a) startswith()** - O método startswith() retorna True se a string começar com o prefixo especificado, caso contrário, retorna False.

**Nível 7:**  
31. **b) open()** - A função open() é usada para abrir um arquivo em Python, retornando um objeto de arquivo.  
32. **b) 'w'** - O modo 'w' abre um arquivo para escrita, criando um novo arquivo se ele não existir ou truncando (apagando o conteúdo) de um arquivo existente.  
33. **b) Uma exceção FileNotFoundError é lançada.** - Se você tentar abrir um arquivo que não existe no modo de leitura, o Python lançará uma exceção FileNotFoundError.  
34. **a) read()** - O método read() lê o conteúdo completo de um arquivo como uma única string.  
35. **d) Utilizar a instrução with open(...) as arquivo:, que fecha o arquivo automaticamente ao sair do bloco with.** - O bloco with garante que o arquivo seja fechado automaticamente, mesmo se ocorrerem exceções dentro do bloco.

**Nível 8:**  
36. **d) Um modelo para criar objetos, definindo seus atributos e métodos.** - Uma classe é como um "blueprint" que descreve a estrutura e o comportamento de seus objetos.  
37. **a) Uma instância de uma classe.** - Um objeto é uma entidade única e independente criada a partir de uma classe, com seus próprios valores de atributo.  
38. **a) \_\_init\_\_()** - O método \_\_init\_\_() é um construtor que é chamado quando um novo objeto da classe é criado, inicializando seus atributos.  
39. **b) A capacidade de uma classe herdar atributos e métodos de outra classe.** - A herança permite criar novas classes (classes filhas) que herdam propriedades e comportamentos de classes existentes (classes pai).  
40. **c) A capacidade de um método ter diferentes implementações em classes diferentes.** - O polimorfismo permite que objetos de classes diferentes sejam tratados de forma genérica, mesmo que respondam a métodos com o mesmo nome de maneiras diferentes.

**Nível 9:**  
41. **a) math** - A biblioteca math fornece funções matemáticas mais complexas, como funções trigonométricas, logaritmos, etc.  
42. **c) pandas** - A biblioteca pandas é uma ferramenta poderosa para manipulação e análise de dados, especialmente para dados em formato tabular.  
43. **d) matplotlib** - A biblioteca matplotlib é amplamente utilizada para criar gráficos estáticos, interativos e animados em Python.  
44. **b) requests** - A biblioteca requests simplifica o processo de fazer requisições HTTP, tornando fácil enviar e receber dados de APIs.  
45. **c) re** - A biblioteca re fornece funções para trabalhar com expressões regulares, permitindo buscar, validar e manipular strings de forma eficiente.

**Nível 10:**  
46. **b) Modificar o comportamento de uma função sem alterá-la diretamente.** - Decoradores são funções que recebem uma função como entrada e retornam uma nova função com funcionalidade adicional.  
47. **b) Uma função que gera uma sequência de valores sob demanda.** - Geradores usam a instrução yield para retornar um valor de cada vez, economizando memória em comparação a retornar uma lista completa de valores.  
48. **a) Multithreading executa múltiplas threads em paralelo em um único processo, enquanto multiprocessing executa múltiplos processos em paralelo.** - A principal diferença está no compartilhamento de recursos: threads compartilham o mesmo espaço de memória do processo, enquanto processos têm seus próprios espaços de memória separados.  
49. **b) Um evento que ocorre durante a execução do programa, interrompendo o fluxo normal.** - Exceções são usadas para lidar com eventos inesperados, como erros de tempo de execução, que impedem o programa de continuar executando normalmente.  
50. **c) Capturar e tratar exceções que podem ocorrer durante a execução de um bloco de código.** - O bloco try contém o código que pode lançar uma exceção, e o bloco except define o código que será executado para lidar com a exceção, evitando que o programa seja interrompido abruptamente.

SQL: Questões 41 a 50

20 questões sobre SQL e MySQL, explorando desde conceitos básicos até desafios de média complexidade, para testar seus conhecimentos nesse universo dos bancos de dados.

**Nível 1: Fundamentos SQL (41-45)**

**41. Qual instrução SQL é utilizada para recuperar dados de uma tabela?**  
a) SELECT  
b) INSERT  
c) UPDATE  
d) DELETE  
e) CREATE

**42. Qual a cláusula utilizada para filtrar os resultados de uma consulta SQL basedo em uma condição específica?**  
a) FROM  
b) WHERE  
c) ORDER BY  
d) GROUP BY  
e) HAVING

**43. Qual o operador SQL utilizado para comparar se dois valores são iguais?**  
a) =  
b) !=  
c) >  
d) <  
e) LIKE

**44. Qual instrução SQL é utilizada para adicionar uma nova coluna a uma tabela existente?**  
a) ADD COLUMN  
b) ALTER TABLE  
c) MODIFY TABLE  
d) UPDATE TABLE  
e) INSERT COLUMN

**45. Qual a finalidade da restrição NOT NULL em uma coluna de tabela?**  
a) Define a coluna como uma chave primária.  
b) Impede que a coluna seja excluída da tabela.  
c) Impede que valores duplicados sejam inseridos na coluna.  
d) Impede que valores nulos sejam inseridos na coluna.  
e) Define um valor padrão para a coluna.

**Nível 2: Explorando Consultas SQL (46-50)**

**46. O que o código SQL a seguir irá retornar?**

SELECT nome FROM clientes WHERE idade > 25;

a) Todos os dados da tabela clientes.  
b) Os nomes de todos os clientes.  
c) Os nomes dos clientes com idade maior que 25 anos.  
d) As idades de todos os clientes.  
e) Um erro, pois a sintaxe da consulta está incorreta.

**47. Qual a cláusula SQL utilizada para ordenar os resultados de uma consulta em ordem decrescente?**  
a) ORDER BY ASC  
b) ORDER BY DESC  
c) SORT BY ASC  
d) SORT BY DESC  
e) REVERSE ORDER

**48. Qual a finalidade da cláusula GROUP BY em uma consulta SQL?**  
a) Filtrar os resultados de uma consulta com base em um grupo específico.  
b) Agrupar linhas com valores iguais em colunas específicas.  
c) Ordenar os resultados de uma consulta por um grupo específico.  
d) Criar um novo grupo de colunas em uma tabela.  
e) Não existe a cláusula GROUP BY em SQL.

**49. O que o código SQL a seguir irá retornar?**

SELECT COUNT(\*) FROM produtos;

a) O número total de colunas na tabela produtos.  
b) O número total de linhas na tabela produtos.  
c) A soma de todos os valores na coluna quantidade da tabela produtos.  
d) A média de todos os valores na coluna preco da tabela produtos.  
e) Um erro, pois a função COUNT(\*) não existe em SQL.

**50. Qual a função SQL utilizada para concatenar strings?**  
a) CONCAT()  
b) JOIN()  
c) MERGE()  
d) APPEND()  
e) SUM()

**Gabarito:**

**Nível 1:**

1. **a) SELECT** - A instrução SELECT é a base para recuperar dados de um banco de dados SQL. Você a utiliza para especificar as colunas e tabelas das quais deseja extrair dados.
2. **b) WHERE** - A cláusula WHERE é crucial para filtrar dados em SQL. Ela permite especificar condições que os dados devem satisfazer para serem incluídos nos resultados da consulta.
3. **a) =** - Em SQL, o operador de igualdade = é usado para comparar se dois valores são iguais. É importante diferenciá-lo da atribuição = utilizada em linguagens de programação como Python.
4. **b) ALTER TABLE** - A instrução ALTER TABLE é uma ferramenta poderosa para modificar a estrutura de uma tabela existente em um banco de dados SQL. Isso inclui adicionar, modificar ou excluir colunas.
5. **d) Impede que valores nulos sejam inseridos na coluna.** - A restrição NOT NULL garante a integridade dos dados, exigindo que um valor válido seja fornecido para a coluna ao inserir ou atualizar dados.

**Nível 2:**

1. **c) Os nomes dos clientes com idade maior que 25 anos.** - A consulta seleciona apenas a coluna nome da tabela clientes, mas aplica um filtro usando a cláusula WHERE idade > 25, garantindo que apenas clientes com mais de 25 anos sejam incluídos no resultado.
2. **b) ORDER BY DESC** - A cláusula ORDER BY permite classificar os resultados da consulta. DESC (abreviação de "descending") é usada para especificar a ordem decrescente, enquanto ASC (abreviação de "ascending") é usada para ordem crescente.
3. **b) Agrupar linhas com valores iguais em colunas específicas.** - A cláusula GROUP BY é frequentemente utilizada com funções agregadas (como COUNT, SUM, AVG) para realizar cálculos em grupos de linhas que compartilham o mesmo valor em uma ou mais colunas.
4. **b) O número total de linhas na tabela produtos.** - A função COUNT(\*) conta o número de linhas na tabela especificada, independentemente de quais colunas são selecionadas.
5. **a) CONCAT()** - A função CONCAT() é amplamente utilizada em SQL para combinar duas ou mais strings em uma única string.

Questões 51 a 60 (+ Gabarito Detalhado)

Com base no sistema de gestão de contas bancárias que discutimos, preparei mais 10 questões de SQL, focando na aplicação prática em um cenário real. As questões abordam desde consultas simples até manipulação de dados, desafiando você a aplicar seus conhecimentos. Vamos lá! 🚀

**Cenário:** Imagine um banco de dados com as seguintes tabelas:

**clientes**:

| **Coluna** | **Tipo de Dado** | **Descrição** |
| --- | --- | --- |
| id | INT | Identificador único do cliente |
| nome | VARCHAR(255) | Nome completo do cliente |
| cpf | VARCHAR(11) | CPF do cliente |
| data\_nascimento | DATE | Data de nascimento do cliente |

**contas**:

| **Coluna** | **Tipo de Dado** | **Descrição** |
| --- | --- | --- |
| id | INT | Identificador único da conta |
| cliente\_id | INT | ID do cliente titular da conta |
| saldo | DECIMAL(10,2) | Saldo atual da conta |
| limite\_credito | DECIMAL(10,2) | Limite de crédito da conta |

**Questões:**

**51. Escreva uma consulta SQL para selecionar o nome e o CPF de todos os clientes.**

**52. Escreva uma consulta SQL para selecionar o ID da conta e o saldo de todas as contas com saldo inferior a R$ 100,00.**

**53. Escreva uma consulta SQL para selecionar o nome do cliente e o saldo da conta, ordenados pelo saldo em ordem decrescente.**

**54. Escreva uma consulta SQL para atualizar o saldo da conta com ID 123 para R$ 500,00.**

**55. Escreva uma consulta SQL para excluir o cliente com ID 456. Certifique-se de que a exclusão seja feita em cascata para a tabela contas, ou seja, a conta do cliente também deve ser excluída.**

**56. Escreva uma consulta SQL para inserir um novo cliente na tabela clientes com os seguintes dados: nome "Maria Silva", CPF "98765432109" e data de nascimento "1990-01-15".**

**57. Escreva uma consulta SQL para selecionar o nome dos clientes que possuem mais de uma conta.**

**58. Escreva uma consulta SQL para calcular o saldo médio de todas as contas.**

**59. Escreva uma consulta SQL para selecionar o nome do cliente e o saldo da conta, mas apenas para clientes maiores de 18 anos. Utilize a data atual para calcular a idade.**

**60. Escreva uma consulta SQL para criar uma nova tabela chamada transacoes com as colunas: id (INT, chave primária), conta\_id (INT), data\_transacao (DATETIME), valor (DECIMAL(10,2)) e tipo (VARCHAR(50)).**