**Classe InsertSQL**

**Visão Geral**

A classe InsertSQL foi projetada para facilitar a interação com bancos de dados MySQL, especificamente para a execução de consultas SQL. Ela oferece métodos para ler configurações de conexão a partir de um arquivo JSON, estabelecer conexões seguras com o banco de dados, executar consultas de inserção (INSERT) e realizar consultas que retornam resultados.

**Construtor**

**\_\_init\_\_(self, query)**

O construtor da classe InsertSQL inicializa a instância com uma consulta SQL (query) que será utilizada pelos métodos da classe.

* **Parâmetros**:
  + query (str): A consulta SQL a ser executada. Pode ser uma instrução SQL de inserção (INSERT), seleção (SELECT), atualização (UPDATE), ou qualquer outra consulta suportada pelo MySQL.

**Métodos**

**read\_connection\_info(self, filename, host, port, user, password, database)**

Este método lê as informações de conexão ao banco de dados a partir de um arquivo JSON. O arquivo JSON deve conter as configurações como host, porta, usuário, senha e nome do banco de dados.

* **Parâmetros**:
  + filename (str): O caminho do arquivo JSON que contém as informações de conexão.
  + host (str): A chave no JSON que corresponde ao host do banco de dados.
  + port (str): A chave no JSON que corresponde à porta do banco de dados.
  + user (str): A chave no JSON que corresponde ao nome de usuário do banco de dados.
  + password (str): A chave no JSON que corresponde à senha do banco de dados.
  + database (str): A chave no JSON que corresponde ao nome do banco de dados.
* **Retorno**:
  + dict: Um dicionário contendo as informações de conexão extraídas do arquivo JSON.
  + None: Retorna None se ocorrer um erro durante a leitura ou decodificação do arquivo JSON.
* **Tratamento de Erros**:
  + O método captura e exibe erros de decodificação de JSON (json.JSONDecodeError), o que ajuda a identificar problemas na estrutura do arquivo JSON.

**connection(self, host, port, user, password, database)**

Este método estabelece uma conexão com o banco de dados MySQL usando as informações obtidas do arquivo JSON. Ele transforma os dados extraídos em um dicionário que pode ser passado diretamente para a função de conexão do mysql.connector.

* **Parâmetros**:
  + host (str): O nome ou endereço IP do host onde o MySQL está rodando.
  + port (str): O número da porta que o MySQL está usando.
  + user (str): O nome de usuário para autenticação no MySQL.
  + password (str): A senha para autenticação no MySQL.
  + database (str): O nome do banco de dados ao qual se deseja conectar.
* **Retorno**:
  + MySQLConnection: Um objeto de conexão MySQL, se a conexão for bem-sucedida.
  + None: Retorna None se ocorrer um erro ao tentar estabelecer a conexão.
* **Tratamento de Erros**:
  + O método captura e exibe erros de conexão do MySQL (mysql.connector.Error), o que pode ser útil para diagnóstico de problemas relacionados à rede, autenticação, ou configurações do banco de dados.

**mysql\_insert(self, host, port, user, password, database)**

Este método executa uma consulta SQL de inserção (INSERT) no banco de dados MySQL. Ele utiliza o método connection para estabelecer a conexão e, em seguida, executa a consulta armazenada no atributo query.

* **Parâmetros**:
  + host (str): O nome ou endereço IP do host onde o MySQL está rodando.
  + port (str): O número da porta que o MySQL está usando.
  + user (str): O nome de usuário para autenticação no MySQL.
  + password (str): A senha para autenticação no MySQL.
  + database (str): O nome do banco de dados ao qual se deseja conectar.
* **Funcionamento**:
  + Estabelece uma conexão com o banco de dados.
  + Cria um cursor, que é um objeto usado para executar as instruções SQL no banco de dados.
  + Executa a consulta SQL INSERT armazenada no atributo query.
  + Realiza o commit das alterações no banco de dados para garantir que a inserção seja permanente.
  + Encerra a conexão com o banco de dados, garantindo a liberação dos recursos.
* **Tratamento de Erros**:
  + Captura e exibe erros durante a execução da consulta SQL (mysql.connector.Error), permitindo que você identifique problemas específicos relacionados à consulta ou à transação.

**mysql\_query(self, host, port, user, password, database)**

Este método executa uma consulta SQL e retorna os resultados. Ele pode ser usado para consultas do tipo SELECT que requerem a recuperação de dados do banco de dados.

* **Parâmetros**:
  + host (str): O nome ou endereço IP do host onde o MySQL está rodando.
  + port (str): O número da porta que o MySQL está usando.
  + user (str): O nome de usuário para autenticação no MySQL.
  + password (str): A senha para autenticação no MySQL.
  + database (str): O nome do banco de dados ao qual se deseja conectar.
* **Retorno**:
  + list: Uma lista contendo os resultados da consulta SQL. Cada item da lista representa uma linha do conjunto de resultados.
* **Funcionamento**:
  + Estabelece uma conexão com o banco de dados.
  + Cria um cursor para executar a consulta SQL.
  + Executa a consulta e usa fetchall para recuperar todos os resultados.
  + Armazena os resultados em uma lista e a retorna ao usuário.
  + Encerra a conexão ao final da operação.
* **Tratamento de Erros**:
  + Captura e exibe erros durante a execução da consulta SQL (mysql.connector.Error), o que pode ser útil para identificar problemas relacionados à sintaxe da consulta ou à integridade dos dados.

**Observações Gerais**

* **Encerramento de Conexões**: Todos os métodos que estabelecem uma conexão com o banco de dados garantem que a conexão seja fechada após a execução das operações, independentemente do sucesso ou falha. Isso é fundamental para evitar vazamentos de recursos e manter a estabilidade do sistema.
* **Manejo de Erros**: O código está preparado para lidar com erros comuns relacionados ao JSON e ao MySQL, o que é crucial para a robustez da aplicação.
* **Extensibilidade**: A classe pode ser facilmente adaptada para incluir novos métodos de manipulação de dados, como atualizações (UPDATE) ou exclusões (DELETE), utilizando a mesma estrutura básica.