**Detalhamento do Projeto: Análise de Dados de Temperatura com Streamlit e PostgreSQL**

Este projeto demonstra um fluxo de trabalho completo para análise de dados, desde a configuração do ambiente até a visualização interativa dos dados em um aplicativo Streamlit hospedado no Render.com. O projeto utiliza um dataset de temperatura, armazena os dados em um banco de dados PostgreSQL e permite a visualização interativa dos dados através de um dashboard Streamlit.

Passo a Passo Detalhado:

Passo 1: Configurar o Repositório no GitHub

1. Criação do Repositório: Um repositório chamado marisabigdatafecaf foi criado no GitHub para armazenar o código do projeto.
2. Conexão Replit-GitHub: O Replit foi conectado ao repositório GitHub para facilitar a sincronização do código.

Passo 2: Configurar o Ambiente no Replit

1. Upload do Dataset: O dataset de temperatura foi baixado e enviado para o Replit.
2. Instalação de Dependências: As bibliotecas necessárias (streamlit, pandas, sqlalchemy, psycopg2-binary, plotly, python-dotenv) foram instaladas usando poetry add.
3. Arquivo .env: Um arquivo .env foi criado para armazenar a string de conexão PostgreSQL (DATABASE\_URL).

Passo 3: Desenvolver o Script em main.py

1. Criação do Script: O arquivo main.py foi criado com o código Streamlit para:
   * Conectar ao banco de dados PostgreSQL.
   * Criar uma tabela no banco de dados a partir do arquivo CSV enviado.
   * Exibir os dados em um gráfico de série temporal usando Plotly.

Passo 4: Executar o Streamlit

1. Execução do Streamlit: O aplicativo Streamlit foi executado usando streamlit run main.py.
2. Upload e Visualização: O arquivo CSV foi enviado através da interface Streamlit, e os dados foram visualizados em um gráfico de série temporal.

Passo 5: Fazer Git Push

1. Adicionar e Comitar Alterações: As alterações no código foram adicionadas e comitadas usando git add . e git commit -m "Adiciona funcionalidade de upload e visualização".
2. Push para o GitHub: As alterações foram enviadas para o repositório GitHub usando git push origin main.

Passo 6: Deploy no Render.com

1. Criação de Conta e Serviço: Uma conta foi criada no Render.com, e um novo serviço web foi criado, conectado ao repositório GitHub.
2. Configuração do Serviço: O comando de inicialização streamlit run main.py foi configurado no Render.com.
3. Deploy: O Render.com construiu e implantou o aplicativo Streamlit.

Capturas de Tela do Dashboard:

* Página de Upload:
* Visualização dos Dados:

Views SQL e seus Propósitos:

1. avg\_temp\_por\_dispositivo: Calcula a média da temperatura para cada dispositivo, permitindo analisar o desempenho médio de cada dispositivo.

SQL

CREATE VIEW avg\_temp\_por\_dispositivo AS

SELECT device\_id, AVG(temperature) as avg\_temp

FROM TEMPERATURA

GROUP BY device\_id;

1. min\_max\_temp\_por\_data: Identifica as temperaturas máxima e mínima registradas em cada data, útil para identificar variações extremas de temperatura ao longo do tempo.

SQL

CREATE VIEW min\_max\_temp\_por\_data AS

SELECT date, MIN(temperature) as min\_temp, MAX(temperature) as max\_temp

FROM TEMPERATURA

GROUP BY date;

1. leituras\_acima\_limite: Conta o número de leituras de temperatura que excedem um determinado limite, útil para identificar dispositivos com leituras anormais.

SQL

CREATE VIEW leituras\_acima\_limite AS

SELECT device\_id, COUNT(\*) as leituras\_acima\_limite

FROM TEMPERATURA

WHERE temperature > 30 -- Substitua 30 pelo limite desejado

GROUP BY device\_id;

Possíveis Insights Obtidos dos Dados:

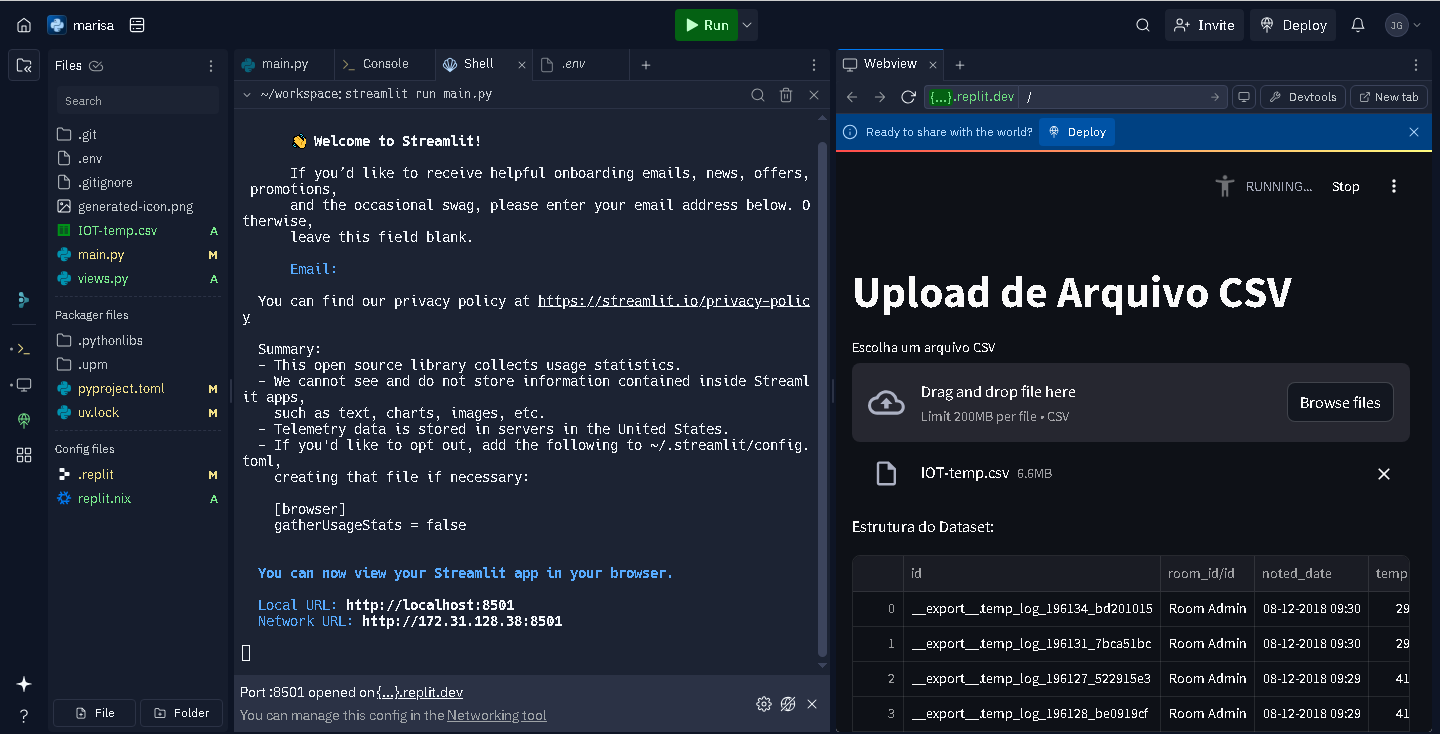
* Variação de Temperatura por Dispositivo: A view avg\_temp\_por\_dispositivo permite identificar dispositivos com temperaturas médias mais altas ou mais baixas, indicando possíveis problemas de desempenho ou localização.
* Variação de Temperatura ao Longo do Tempo: A view min\_max\_temp\_por\_data permite identificar padrões de variação de temperatura ao longo do tempo, como picos de temperatura em determinados horários ou dias.
* Identificação de Leituras Anormais: A view leituras\_acima\_limite permite identificar dispositivos com leituras de temperatura acima de um limite predefinido, indicando possíveis problemas de funcionamento ou necessidade de manutenção.

Considerações Finais:

* Banco de Dados: Certifique-se de que o banco de dados PostgreSQL esteja acessível a partir do ambiente onde o Streamlit está sendo executado.
* Segurança: Nunca exponha credenciais sensíveis diretamente no código. Use variáveis de ambiente para gerenciá-las.
* Debugging: Se encontrar erros, verifique os logs no Replit e no Render.com para diagnósticos.

Este projeto demonstra um fluxo de trabalho completo para análise de dados, desde a configuração do ambiente até a visualização interativa dos dados em um aplicativo Streamlit hospedado no Render.com.

Imagens práticas:



Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.