backend\eda_logic.py

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import ollama
import os

# Function to Perform EDA and Generate Visualizations
def eda_analysis(file_path, output_dir="static"):
    df = pd.read_csv(file_path)

    # Fill missing values with median for numeric columns
    for col in df.select_dtypes(include=['number']).columns:
        df[col].fillna(df[col].median(), inplace=True)

    # Fill missing values with mode for categorical columns
    for col in df.select_dtypes(include=['object']).columns:
        df[col].fillna(df[col].mode()[0], inplace=True)

    # Data Summary
    summary = df.describe(include='all').to_string()

    # Missing Values
    missing_values = df.isnull().sum().to_string()

    # Generate AI Insights
    insights = generate_ai_insights(summary)

    # Generate Data Visualizations
    plot_paths = generate_visualizations(df, output_dir)

    return {
        "summary": summary,
        "missing_values": missing_values,
        "insights": insights,
        "plot_paths": plot_paths
    }

# AI-Powered Insights using gemma3:270m (Ollama)
def generate_ai_insights(df_summary):
    prompt = f"Analyze the dataset summary below. \n1. Identify the likely Dependent Variable (Target) and Independent Variables (Features).\n2. Provide key insights and trends.\n\nSummary:\n{df_summary}"
    try:
        # Using a model available in Ollama (ensure user has this or change to a default like 'llama3' or 'mistral')
        # The original code used "gemma3:270m"
        response = ollama.chat(model="gemma3:270m", messages=[{"role": "user", "content": prompt}])
```

```
45          return response['message']['content']
46      except Exception as e:
47          return f"Error generating AI insights: {str(e)}"
48
49  # Function to Generate Data Visualizations
50  def generate_visualizations(df, output_dir):
51      if not os.path.exists(output_dir):
52          os.makedirs(output_dir)
53
54      plot_paths = []
55
56      # Histograms for Numeric Columns
57      for col in df.select_dtypes(include=['number']).columns:
58          plt.figure(figsize=(6,4))
59          sns.histplot(df[col], bins=30, kde=True, color="blue")
60          plt.title(f"Distribution of {col}")
61
62          filename = f"{col}_distribution.png"
63          path = os.path.join(output_dir, filename)
64
65          plt.savefig(path)
66          plot_paths.append(filename) # Return just the filename for static serving
67          plt.close()
68
69      # Correlation Heatmap (only numeric columns)
70      numeric_df = df.select_dtypes(include=['number'])
71      if not numeric_df.empty:
72          plt.figure(figsize=(8,5))
73          sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
74          plt.title("Correlation Heatmap")
75
76          filename = "correlation_heatmap.png"
77          path = os.path.join(output_dir, filename)
78
79          plt.savefig(path)
80          plot_paths.append(filename)
81          plt.close()
82
83      return plot_paths
84
85  def ask_ai_question(question, context):
86      prompt = f"Context (Dataset Summary):\n{context}\n\nQuestion: {question}\n\nAnswer:"
87      try:
88          response = ollama.chat(model="gemma3:270m", messages=[{"role": "user", "content":
    prompt}])
89          return response['message']['content']
90      except Exception as e:
91          return f"Error connecting to AI: {str(e)}"
92
```