Initially

🧠💻 Project: Build Your Own LLM-Based Data Analyst (From Scratch)

🧠🛠️ Reframed Project: Build a Rule-Based + ML Data Analyst (No Transformers)

Let’s break this down into a classic ML + rules + heuristics + stats pipeline — no LLMs or deep learning required.

✅ What You'll Build (No Transformers):

* Upload CSV or connect to DB
* Ask natural language questions
* Parse + interpret the question using classic NLP
* Use rules / simple ML to find what to do (e.g., aggregation, filtering)
* Generate answers (text) + visualizations

Think of this as building your own mini Tableau with a chatbot interface, powered by:

* Rule-based NLP
* Text classification
* Heuristics
* Statistical reasoning

🧠 How to Do It Without Transformers

🧩 1. NLP Pipeline (Question Understanding)

Use lightweight NLP:

* spaCy or nltk to tokenize, extract nouns, verbs
* Simple keyword classification:
  + if "average", "mean", → it's an aggregation
  + if "trend", "over time" → it's a time series plot
  + if "top", "highest", "most" → it's sorting
* Optional: train a small classifier (e.g., logistic regression) to classify question types:
  + Aggregation
  + Comparison
  + Filter + display
  + Plot request

🧪 2. Table Profiler

Create a table summary:

* List columns + types
* Find date columns
* Check for numerical / categorical fields
* Sample a few rows

This will help match the user's question with the relevant columns.

💡 3. Intent Matching Logic (No LLM)

Create a rule-based matching engine that maps:

* User intent (average, count, trend)
* To column(s) (via fuzzy matching or keyword overlap)

Example:

# user says: "Show me the trend of sales over the last year"

# intent = "line\_chart", metric = "sales", time = "year"

if "trend" in question and "over" in question:

chart\_type = "line"

Use fuzzywuzzy, rapidfuzz, or difflib.get\_close\_matches to match column names with question keywords.

📊 4. Chart + Answer Generator

Based on the parsed intent:

* Run a pandas query
* Generate a chart with matplotlib or plotly
* Build a templated textual answer:
  + “The average sales in Q4 was $12,400.”

🖼️ 5. UI (Streamlit or CLI)

* Upload file
* Ask a question
* See:
  + Parsed intent
  + Data preview
  + Text + chart output

✨ Bonus: Add ML for Classification

Train a simple model on synthetic Q&A pairs (just like the transformer version):

* Use TF-IDF + Logistic Regression / Naive Bayes
* Train on question → intent type (avg, top-k, filter, plot)
* Train on question → target column (multi-label)

Still fast, still interpretable.

🔧 Tools to Use

| Task | Tool / Lib |
| --- | --- |
| NLP | spaCy, nltk, re |
| Column matching | fuzzywuzzy, difflib |
| ML model (optional) | scikit-learn |
| Visualizations | matplotlib, seaborn, plotly |
| UI | streamlit / CLI |

📁 Suggested Structure (No Transformers Version)

classic-data-analyst/

├── app/

│ └── streamlit\_app.py

├── core/

│ ├── intent\_parser.py # rule-based NLP

│ ├── column\_matcher.py # fuzzy matching

│ ├── analyzer.py # run pandas queries

│ └── viz\_generator.py

├── data/

│ └── sample\_datasets/

├── models/

│ └── (optional) scikit models

├── requirements.txt

└── README.md

✅ Summary

| Version | Pros | Cons |
| --- | --- | --- |
| Transformer version | More flexible, natural understanding | Requires compute + fine-tuning |
| Rule-based + ML (this) | Lightweight, fast, transparent | Needs more manual rule tuning |

Ready to Start?

I can help you:

* 🧠 Draft the intent detection rules
* ✍️ Write a fuzzy column matcher
* ⚙️ Build a question → action → chart pipeline

Want me to help scaffold the first couple modules — like intent\_parser.py or a sample dataset + question pair?

Absolutely — here’s the full **workflow for your LLM-free, lightweight Data Analyst app**, with a clear **step-by-step pipeline** from user input to visual output.

**🧠💻 Project Workflow: Rule-Based + ML-Powered Data Analyst (No Transformers)**

**🧭 Objective:**

Allow a user to upload a CSV and ask natural language questions like:

* “What’s the average salary?”
* “Show sales trend over the last year”
* “Top 5 departments by headcount”
* “How many employees joined in Q1 2023?”

You’ll process that input using **traditional NLP, rule-based logic, and light ML** to generate:

* Textual answers
* Visualizations (charts)

**🔄 High-Level Workflow**

[User Input: CSV + Question]

|

↓

1️⃣ File Ingestion + Table Profiling

|

↓

2️⃣ Question Preprocessing + Intent Detection

|

↓

3️⃣ Column Matching + Context Building

|

↓

4️⃣ Data Query Execution (Pandas)

|

↓

5️⃣ Text Answer Generator

|

↓

6️⃣ Chart Generator (matplotlib/plotly)

|

↓

7️⃣ Output: Text + Visualization in UI

**🧱 Step-by-Step Breakdown**

**1️⃣ File Ingestion + Table Profiling**

* Use pandas to load the file
* Extract:
  + Column names
  + Data types (numerical, categorical, datetime)
  + Summary stats (min, max, nulls, uniques)
* Store this as a table\_profile object

📦 **Example:**

{

"columns": ["name", "department", "salary", "hire\_date"],

"types": {"salary": "numeric", "hire\_date": "datetime"},

"sample\_rows": [...]

}

**2️⃣ Question Preprocessing + Intent Detection**

* Lowercase, clean text (re, nltk, or spaCy)
* Detect:
  + **Action**: average, sum, count, show, plot, trend, filter
  + **Metric**: what column(s) are mentioned?
  + **Condition**: time frame? filters?

📦 **Output:**

{

"intent": "aggregation",

"aggregation": "mean",

"target\_column": "salary",

"filters": None

}

You can use rule-based templates or a trained TF-IDF + LogisticRegression model to classify intent.

**3️⃣ Column Matching**

Match target words from question to column names using:

* fuzzywuzzy or difflib.get\_close\_matches
* Token-level matching (lemmatized)
* Context-aware mapping (e.g., “hiring date” → “hire\_date”)

📦 **Example:**

"salary" → "salary" (exact match)

"sales" → "sales\_amount" (fuzzy match score: 92)

"joined" → "hire\_date" (via verb mapping)

**4️⃣ Data Query Execution**

Use pandas to:

* Filter, group, aggregate
* Time window slicing (for trend questions)
* Sort and get top-k

📦 **Example Query:**

df.groupby("department")["employee\_id"].count().sort\_values(ascending=False).head(5)

**5️⃣ Text Answer Generator**

Generate templated text like:

“The average salary is $73,500.”  
“Top 3 departments by headcount are Sales, Engineering, and HR.”

Use simple string templates based on intent + query result.

📦 Example:

f"The average {target\_column} is {result:.2f}"

**6️⃣ Chart Generator**

Based on intent, generate visualizations:

* line → trend over time
* bar → top-k
* pie → proportion
* hist → distribution

Use plotly or matplotlib to generate charts dynamically.

📦 Chart Builder Template:

import matplotlib.pyplot as plt

def plot\_top\_k(data, column, k=5):

top = data[column].value\_counts().head(k)

top.plot(kind="bar")

**7️⃣ Output in Streamlit**

* Display:
  + File preview
  + Parsed question analysis
  + Text answer
  + Chart
* Optionally log the Q&A pairs

**🧠 Bonus (Optional Layers)**

| **Feature** | **How to Add** |
| --- | --- |
| User feedback | Add thumbs up/down + retraining data |
| Intent model | Train scikit-learn model to classify Q types |
| Explanation panel | Show parsed intent, matching logic, Pandas query preview |
| Export | Allow download of query result or analysis PDF |
| Multi-turn chat | Store conversation state, pass previous Q&A as context |

**🧪 Testing Suggestions**

Build a test set of 20–30 realistic questions over 3–5 datasets:

* HR data (employees, salaries)
* Sales data (products, revenue)
* Customer data (churn, demographics)

Write assertions like:

assert parse\_intent("average salary") == {"intent": "aggregation", "agg": "mean"}

assert column\_match("revenue", ["sales", "revenue", "profit"]) == "revenue"

**✅ Ready to Start?**

Here’s a suggested starting point:

**Phase 1: Build the Core Modules**

* table\_profiler.py → handles CSV loading and data typing
* intent\_parser.py → rule-based intent + keyword extractor
* column\_matcher.py → fuzzy map question terms to columns
* query\_runner.py → builds and runs pandas operations
* viz\_generator.py → selects chart type + plots

Let me know and I’ll scaffold these files for you. Or do you want to start with one specific part — like intent\_parser.py or streamlit\_app.py?