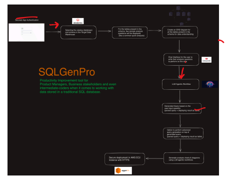
WHO? → PM, Business, non-coders, code
WHAT? → Text → to SQL
WHY? → clarity
HOW? → How are they going to use?

1) Text → Engine (LLM) → SQL (core)
2) Take user input (Text)
3) Use Engine to Create SQL → error
4) Validate & Fix SQL (by execution) correct
5) Output generated SQL
6) Option to get results
7) Secure → ① conditional → version, ② → HTTPS/.



Challengs with Text-to-SQL:
1) Multiple tables
2) multiple columns per table
3) Categorical columns
} But limited context length

Good Context is Key to reduce HALLUCINATION

Prompt (Question, Context, Example) → LLM → Output (SQL) → Validate
(i) Syntax
(ii) column reference
By SQL Execution method.
Validate → incorrect → (back to Prompt)
Validate → correct → To user

For Text-to-sql.
(i) Table schema
(ii) Sample rows
(iii) Additional enrichments

Challengs with Text-to-SQL:
1) Multiple tables
2) multiple columns per table
3) Categorical columns
} But limited context length

Adding Favourites

Save → Question / query → Insert to table

Favourite Section → click → List all Previously saved questions → select → Retrieve the query from table

Advantages:
1) Cost efficient
2) Deterministic results
3) Less user work

Quick Analysis:

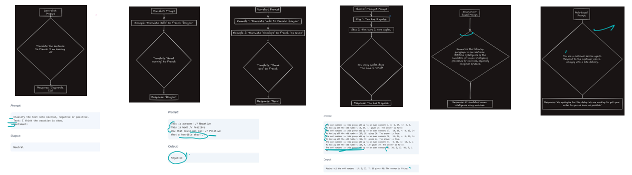
Right question → Data → Right Answer

Advantags
1) Inspiration
2) Proactive analysis

Prompt Engineering:

1) LLM settings:
- Temperature
- Top P
- max length
- Stop Sequences
- Frequency penalty
- Presence Penalty

2) Prompting Techniques
Instruction, Context, Input, Output



Langchain:

LlamaIndex

LangChain is a framework designed to simplify the process of building applications powered by Large Language Models (LLMs). It provides tools for chaining multiple prompts together, handling input/output processing, managing API calls, and more. Essentially, LangChain allows developers to create more complex, intelligent applications that can handle tasks like dialogue management, context retention, decision-making, and more by combining the power of LLMs with additional logic and context.

Key Features of LangChain:

- **Prompt Chaining**: Allows you to combine multiple prompts into a cohesive workflow.
- **Memory Management**: Helps manage context between different prompts, enabling conversations or tasks that require long-term context retention.
- **Tools and APIs**: Integrates with various APIs and tools to extend the functionality of the LLMs.
- **Data Parsing**: Offers utilities to parse and structure the outputs from LLMs.

chaining: P1 → LLM → output → P2 → LLM → Final result

Memory for conversation: P → LLM {Memory} → output
Input + output Pairs

Tools: P → LLM → output
LLM ↔ Internet
LLM ↔ weather API

Dataparsing: P → LLM → Json output { }