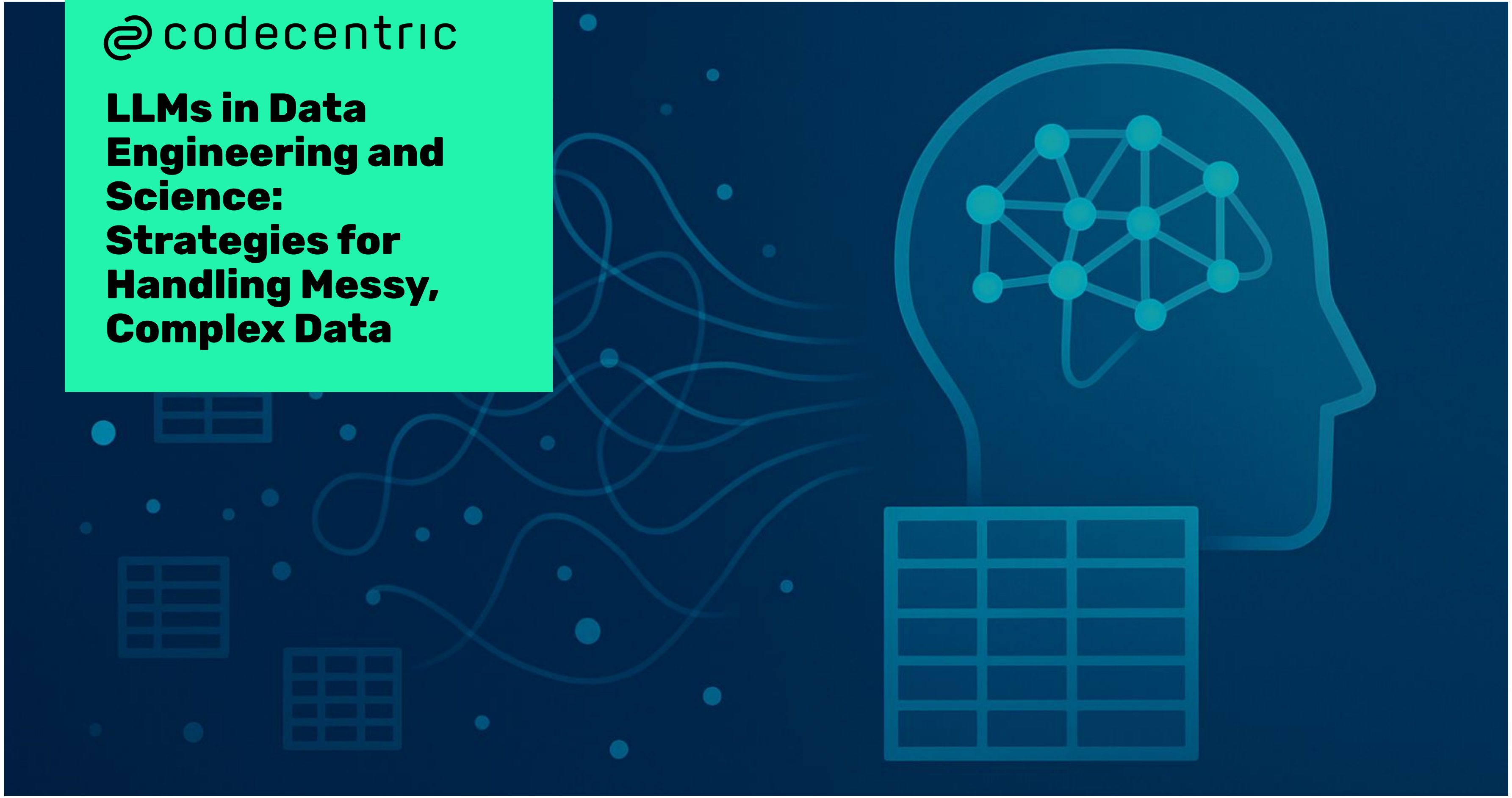
codecentric

LLMs in Data Engineering and Science: Strategies for Handling Messy, Complex Data

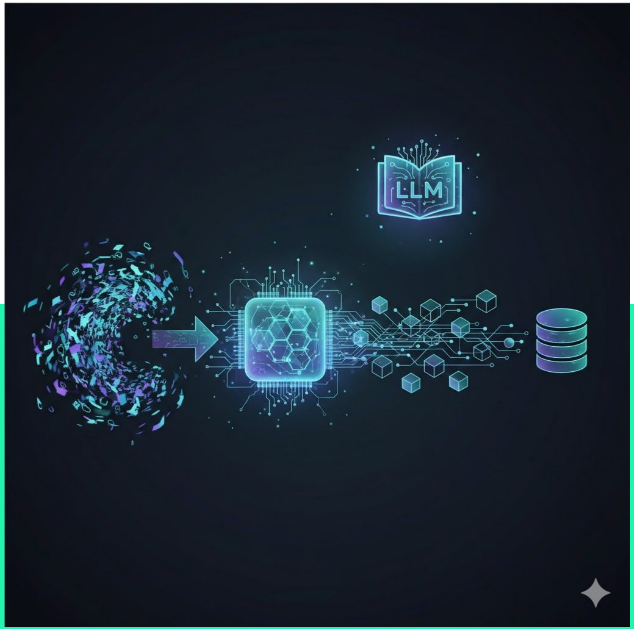


Agenda

Use Case: Data Transformation

Use Case: Data Science Buddy

Conclusion



Data Transformation

First Use Case: Data Transformation

Project Brief

Retail Customer

Wants to expand from a few thousand, to a million products

Marketplace structure (others sell on their platform)

Bad Input Data

CSV with messy format:

- Different manufacturers have different attributes
- even manufacturer is not consistent
- one column can contain multiple values

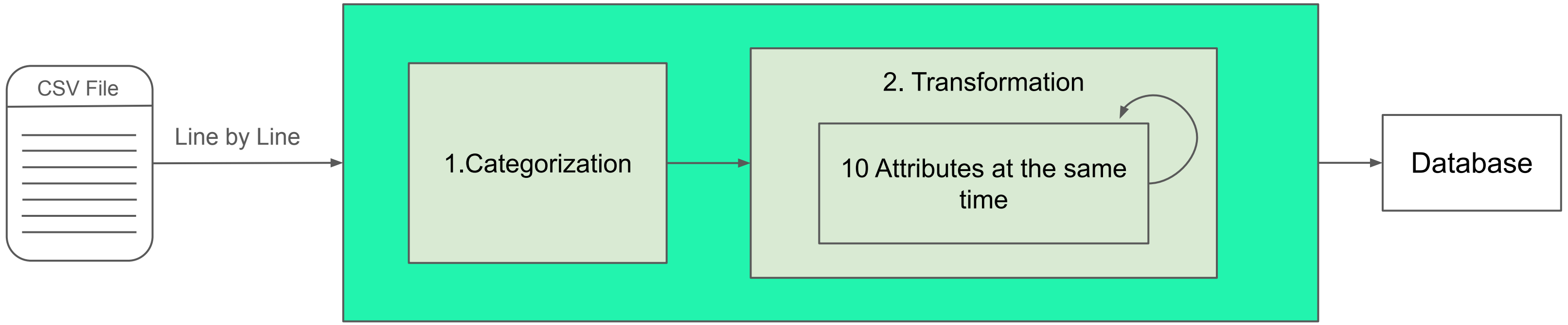
Output Format Complex

Internal Product Structure:

- around 30k attributes
- depending on category 1-3k relevant
- In the end, product fills 100-300
- One attribute might need multiple columns

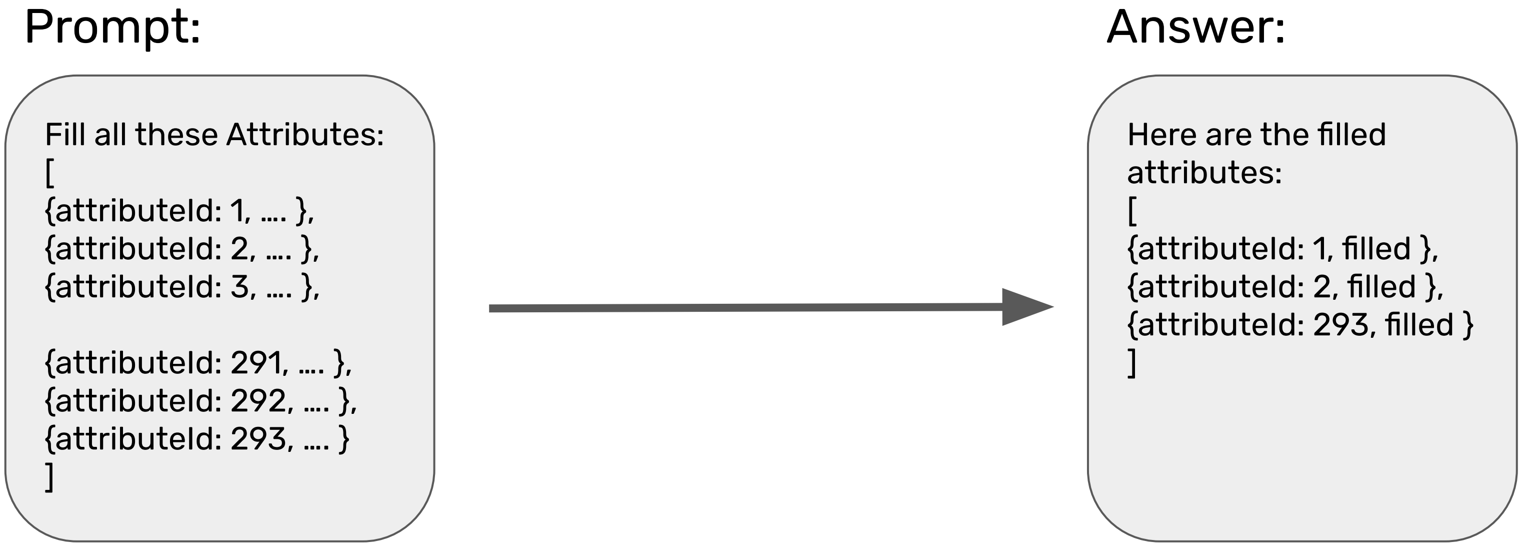
First Use Case: Data Transformation

Our Architecture - LLM Based



First Use Case: Data Transformation

First Lesson - Don´t trust Context length



Prompt:

```
Fill all these Attributes:
[
{attributeId: 1, …. },
{attributeId: 2, …. },
{attributeId: 3, …. },

{attributeId: 291, …. },
{attributeId: 292, …. },
{attributeId: 293, …. }
]
```

Answer:

```
Here are the filled
attributes:
[
{attributeId: 1, filled },
{attributeId: 2, filled },
{attributeId: 293, filled }
]
```

First Use Case: Data Transformation

Improvement: Chain of Thought Prompting

Prompt:

```
Fill all these Attributes
and state the next
attribute to fill:
[
{attributeId: 1, …. },
{attributeId: 2, …. },
{attributeId: 3, …. },

{attributeId: 291, …. },
{attributeId: 292, …. },
{attributeId: 293, …. }
]
```

Answer:

```
Here are the filled attributes:
[
{attributeId: 1, filled, next: 2 },
{attributeId: 2, filled, next: 3 },
{attributeId: 3, filled, next: 4 },
{attributeId: 4, filled, next: 5 },
{attributeId: 5, filled, next: 6 },
{attributeId: 6, filled, next: 7 },
{attributeId: 7, filled, next: 8 },
]
```

Better! But not perfect!

First Use Case: Data Transformation

Improvement: Iterative

Prompt:

Fill all these Attributes
and state the next
attribute to fill:
[
{attributeId: 1, …. },
{attributeId: 2, …. },
{attributeId: 3, …. },
…
{attributeId: 10, …. },
]

Answer:

Here are the filled attributes:
[
{attributeId: 1, filled, next: 2 },
{attributeId: 2, filled, next: 3 },
{attributeId: 3, filled, next: 4 },
{attributeId: 4, filled, next: 5 },
{attributeId: 5, filled, next: 6 },
…
{attributeId: 10, filled, next: none },
]

First Use Case: Data Transformation

Second Lesson - Use Structured Output

OpenAI LLMs can return output to given structure

- **More consistent than few shot prompting**
  - **not one error in our extensive trials**
- **Allows for nice code**

Output Format

attribute_id: int
name: str
source: str (**Beschreibung**: "Gib hier die Stelle aus der CSV wieder, die die nötige Information enthält, die den value bestimmt.")
found_in_source: bool
value: str | bool | float | int (**Beschreibung**: Der Ergebniswert. Der Wert sollte sich aus dem source Attribut ergeben. Steht in der source, dass die Info in der Produktinformation nicht vorhanden ist, so schreibe immer 'None'.)

Third Lesson - Chain of Thought Prompting

LLMs are inconsistent

- Obviously reruns are different
- Also true for fact based attributes
- sometimes uses world knowledge, sometimes not

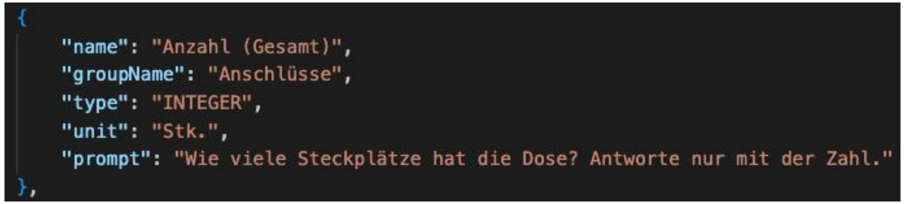
attribute_id: int
name: str
source: str (**Beschreibung**: "Gib hier die Stelle aus der CSV wieder, die die nötige Information enthält, die den value bestimmt.")
found_in_source: bool
value: str | bool | float | int (**Beschreibung**: Der Ergebniswert. Der Wert sollte sich aus dem source Attribut ergeben. Steht in der source, dass die Info in der Produktinformation nicht vorhanden ist, so schreibe immer 'None'.)

How we improved:

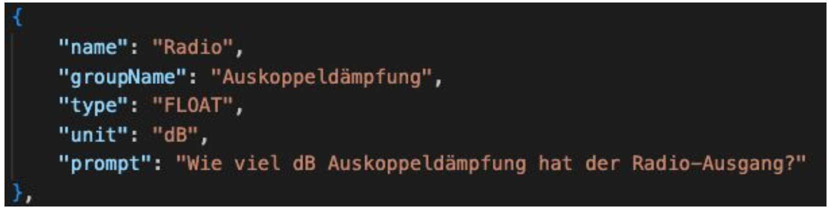
- LLM should only use the source!
- Improved by forcing to list the source
- Further improved to write factual conclusion before

Fourth Advice - Add Metadata

Your own Data is not readable



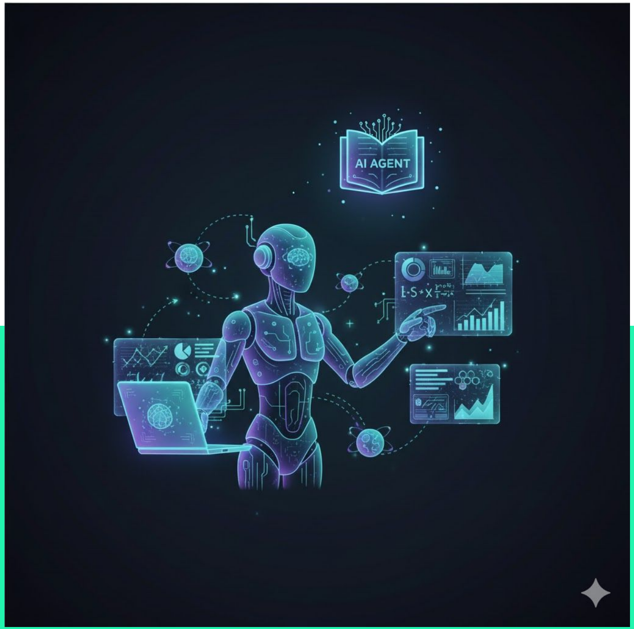
```
{
    "name": "Anzahl (Gesamt)",
    "groupName": "Anschlüsse",
    "type": "INTEGER",
    "unit": "Stk.",
    "prompt": "Wie viele Steckplätze hat die Dose? Antworte nur mit der Zahl."
},
```



```
{
    "name": "Radio",
    "groupName": "Auskoppeldämpfung",
    "type": "FLOAT",
    "unit": "dB",
    "prompt": "Wie viel dB Auskoppeldämpfung hat der Radio-Ausgang?"
},
```

How we improved:

- **Each attribute gets a description parameter**
  - **Each attribute got their own “prompt”**
- **Cross Domain effort**



Data Science Buddy

Second Use Case: Data Transformation

Project Brief

Fashion Industry

Certain products underperformed this year

Why?
Maybe AI can give a different view

Huge Data Treasure

Proud of the data

- They can identify which marketing material led to buy decision
- leads to over 7000 tables
- Only known by deep experts

Two Approaches

Fast:

- Use Claude Code to answer theories

Slow:

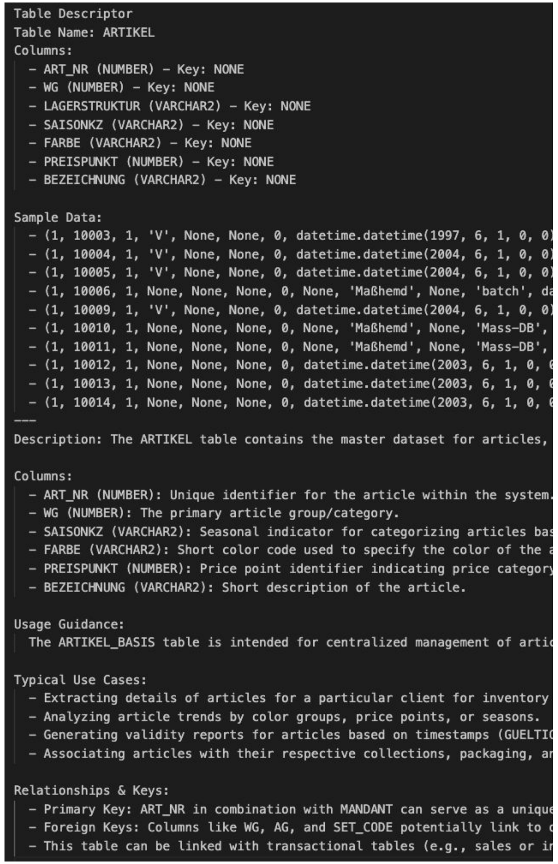
- Build own Agent System

Second Use Case: Data Transformation

Initial Setup

Add metadata to the database

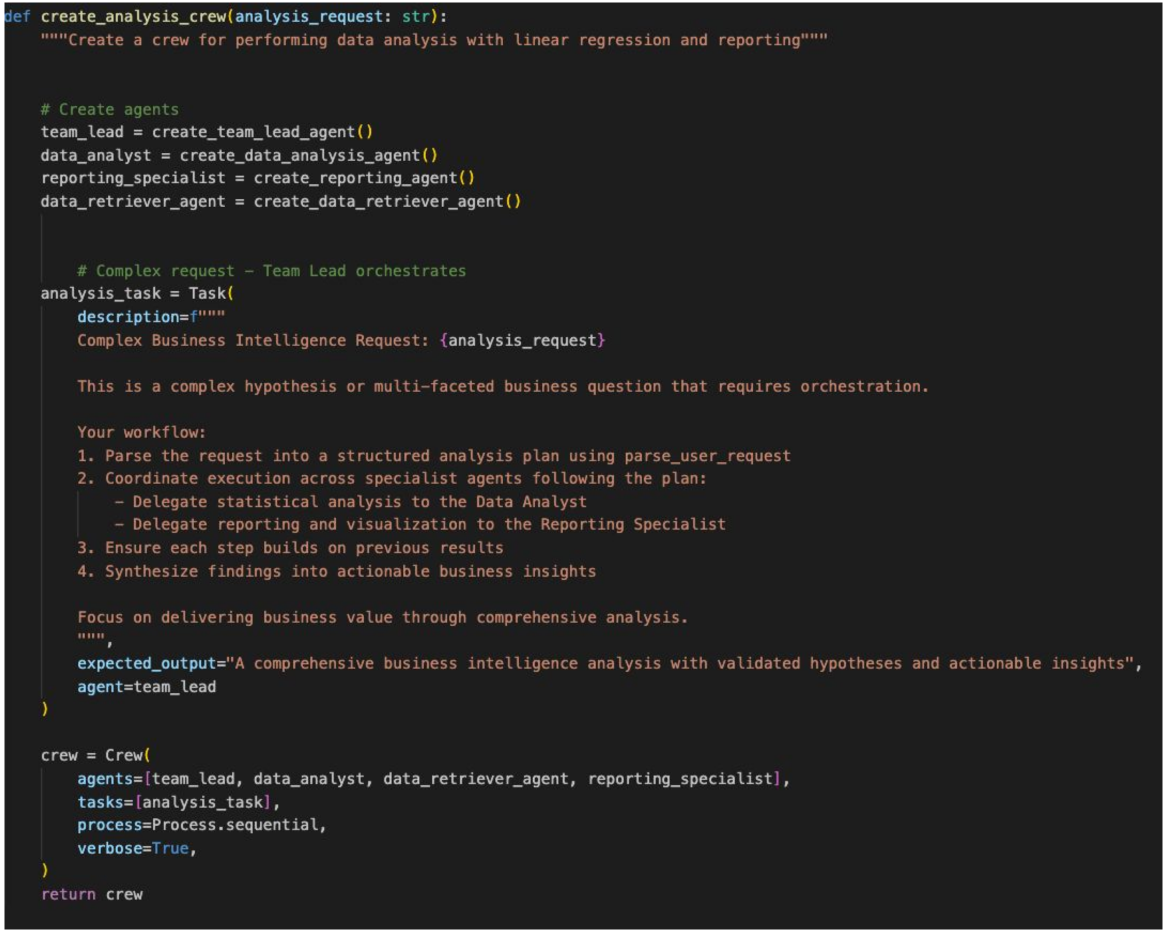
- we identified relevant tables beforehand
- added descriptors to the tables and columns
- let LLM sample the data
- Generate an .md file with those information
- Let LLM fill in some conclusions about the table



```
Table Descriptor
Table Name: ARTIKEL
Columns:
  - ART_NR (NUMBER) - Key: NONE
  - WG (NUMBER) - Key: NONE
  - LAGERSTRUKTUR (VARCHAR2) - Key: NONE
  - SAISONKZ (VARCHAR2) - Key: NONE
  - FARBE (VARCHAR2) - Key: NONE
  - PREISPUNKT (NUMBER) - Key: NONE
  - BEZEICHNUNG (VARCHAR2) - Key: NONE

Sample Data:
  - (1, 10003, 1, 'V', None, None, 0, datetime.datetime(1997, 6, 1, 0, 0)
  - (1, 10004, 1, 'V', None, None, 0, datetime.datetime(2004, 6, 1, 0, 0)
  - (1, 10005, 1, 'V', None, None, 0, datetime.datetime(2004, 6, 1, 0, 0)
  - (1, 10006, 1, None, None, None, 0, None, 'Maßhemd', None, 'batch', da
  - (1, 10009, 1, 'V', None, None, 0, datetime.datetime(2004, 6, 1, 0, 0)
  - (1, 10010, 1, None, None, None, 0, None, 'Maßhemd', None, 'Mass-DB',
  - (1, 10011, 1, None, None, None, 0, None, 'Maßhemd', None, 'Mass-DB',
  - (1, 10012, 1, None, None, None, 0, datetime.datetime(2003, 6, 1, 0, 0
  - (1, 10013, 1, None, None, None, 0, datetime.datetime(2003, 6, 1, 0, 0
  - (1, 10014, 1, None, None, None, 0, datetime.datetime(2003, 6, 1, 0, 0
---
Description: The ARTIKEL table contains the master dataset for articles,

Columns:
  - ART_NR (NUMBER): Unique identifier for the article within the system.
  - WG (NUMBER): The primary article group/category.
  - SAISONKZ (VARCHAR2): Seasonal indicator for categorizing articles bas
  - FARBE (VARCHAR2): Short color code used to specify the color of the a
  - PREISPUNKT (NUMBER): Price point identifier indicating price category
  - BEZEICHNUNG (VARCHAR2): Short description of the article.

Usage Guidance:
  The ARTIKEL_BASIS table is intended for centralized management of artic

Typical Use Cases:
  - Extracting details of articles for a particular client for inventory
  - Analyzing article trends by color groups, price points, or seasons.
  - Generating validity reports for articles based on timestamps (GUELTIC
  - Associating articles with their respective collections, packaging, ar

Relationships & Keys:
  - Primary Key: ART_NR in combination with MANDANT can serve as a unique
  - Foreign Keys: Columns like WG, AG, and SET_CODE potentially link to c
  - This table can be linked with transactional tables (e.g., sales or ir
```

Second Use Case: Data Transformation

MultiAgent Setup



```python
def create_analysis_crew(analysis_request: str):
    """Create a crew for performing data analysis with linear regression and reporting"""


    # Create agents
    team_lead = create_team_lead_agent()
    data_analyst = create_data_analysis_agent()
    reporting_specialist = create_reporting_agent()
    data_retriever_agent = create_data_retriever_agent()


        # Complex request - Team Lead orchestrates
    analysis_task = Task(
        description=f"""
        Complex Business Intelligence Request: {analysis_request}

        This is a complex hypothesis or multi-faceted business question that requires orchestration.

        Your workflow:
        1. Parse the request into a structured analysis plan using parse_user_request
        2. Coordinate execution across specialist agents following the plan:
            - Delegate statistical analysis to the Data Analyst
            - Delegate reporting and visualization to the Reporting Specialist
        3. Ensure each step builds on previous results
        4. Synthesize findings into actionable business insights

        Focus on delivering business value through comprehensive analysis.
        """,
        expected_output="A comprehensive business intelligence analysis with validated hypotheses and actionable insights",
        agent=team_lead
    )

    crew = Crew(
        agents=[team_lead, data_analyst, data_retriever_agent, reporting_specialist],
        tasks=[analysis_task],
        process=Process.sequential,
        verbose=True,
    )
    return crew
```

Slow Expirement

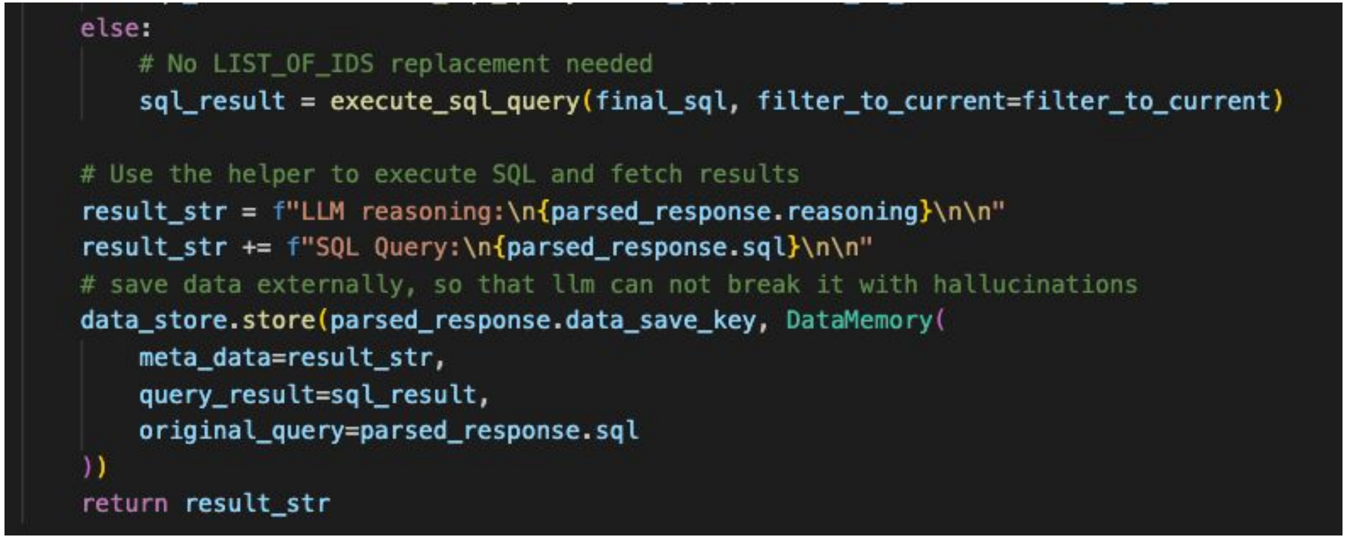
Very Complex

- Data Science methods need to be extremely flexible
- Debugging very difficult
- Time Consuming to self implement

Second Use Case: Data Transformation

First Lesson: Keep Data away from LLM

- **Keep data hallucination free**
  - **Don´t let LLM get their hands on the data**
  - **use different storage and LLM can direct it to the analysis**

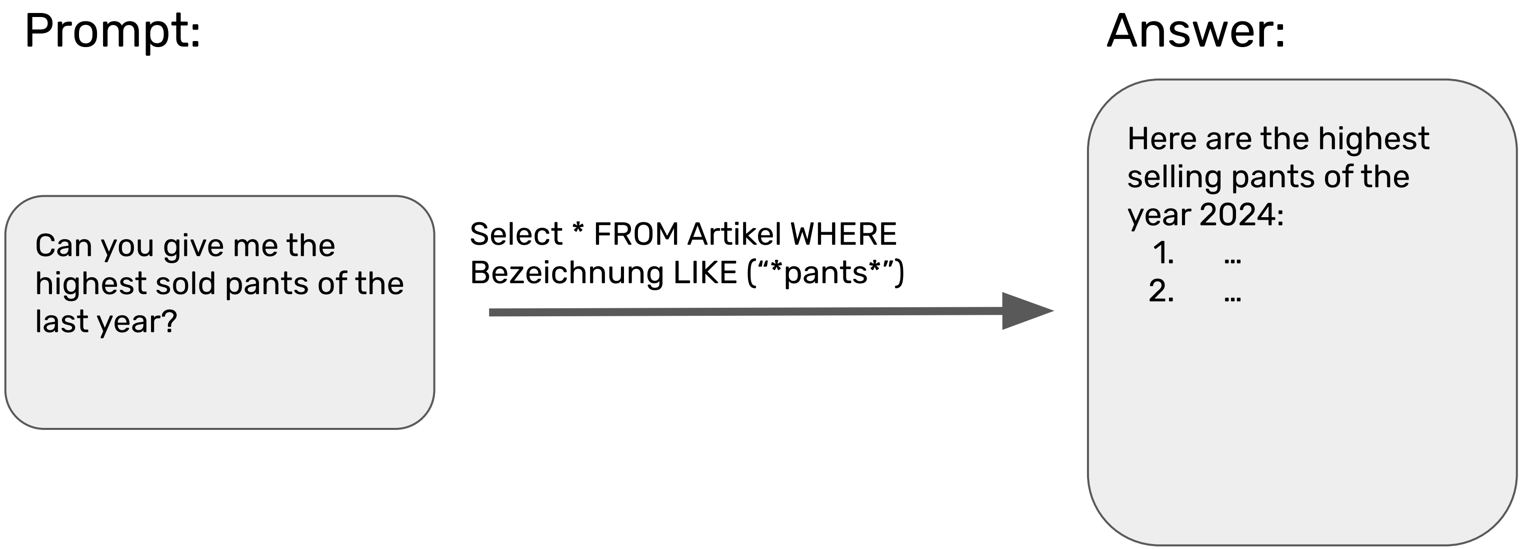


```
else:
    # No LIST_OF_IDS replacement needed
    sql_result = execute_sql_query(final_sql, filter_to_current=filter_to_current)

# Use the helper to execute SQL and fetch results
result_str = f"LLM reasoning:\n{parsed_response.reasoning}\n\n"
result_str += f"SQL Query:\n{parsed_response.sql}\n\n"
# save data externally, so that llm can not break it with hallucinations
data_store.store(parsed_response.data_save_key, DataMemory(
    meta_data=result_str,
    query_result=sql_result,
    original_query=parsed_response.sql
))
return result_str
```

codecentric

Second Use Case: Data Transformation

Second Lesson: Teach Agents Processes



Prompt:

Can you give me the highest sold pants of the last year?

```
Select * FROM Artikel WHERE
Bezeichnung LIKE ("*pants*")
```

Answer:

Here are the highest selling pants of the year 2024:

1. …
2. …

Second Use Case: Data Transformation

Second Lesson: Teach Agents Processes

Prompt:

Can you give me the highest sold pants of the last year?

Select * FROM Artikel WHERE Bezeichnung LIKE ("*pants*")

Answer:

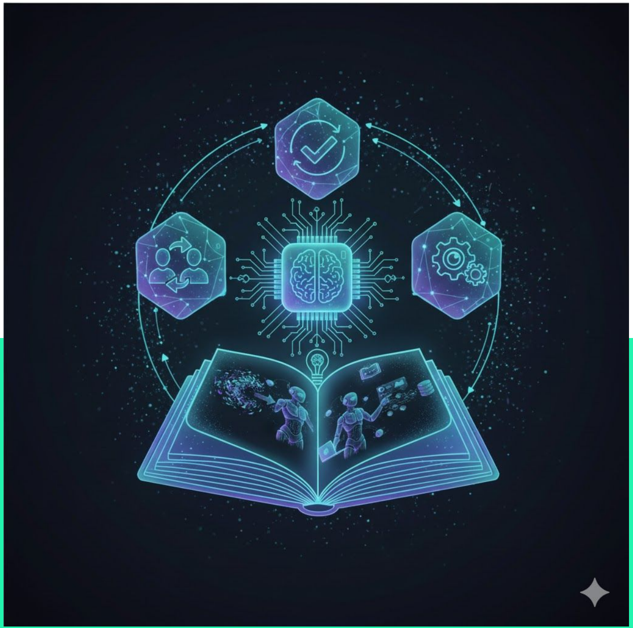
Here are the highest selling pants of the year 2024:

1. …
2. …

Jeans are also Pants!

Second Lesson: Teach Agents Processes

- **Implement strict processes**
  - **Via prompting**
  - **Via Code**



Conclusion

Lessons Learned

Context Engineering

Keep context small

Structured Output

Lessons Learned

Chain of Thought

Reproducible Steps

Free Flow vs Process

codecentric

codecentric AG
Hochstraße 11
42697

Daniel Töws
IT-Consultant
daniel.toews@codecentric.de
Linkedin:
https://www.linkedin.com/in/danieltoews91/