## 1 Methods

To investigate the potential of our proposed approach, we designed a study centered around a webshop interface that enables users to directly compare dynamically generated SQL queries with traditional static templates [**?** ]. This dual-display setup allows participants to experience both methods in real time and assess their effectiveness in retrieving relevant search results. To ensure a focused and controlled analysis, we limited the webshop to a single product category. We chose dogs as the focal category due to their popularity and the diverse range of attributes—such as breed, age, gender, color, size, and over 50 unique features—that lend themselves well to complex, semantic search queries. The underlying dataset comprises 147 AI-generated dog records, synthesized using ChatGPT and Claude, and enriched with real-world data from www.edogs.de [**?** ]. A structured relational database was developed to support efficient querying across these attributes.

The interface presents search results in two side-by-side sections, as illustrated in Fig. **??**. The left section displays results from AI-driven, dynamically generated queries, while the right section shows results from a static query template enhanced with semantic matching. Users submit their queries via a shared search bar, with filtering options applied consistently across both sections through a unified sidebar. Although the AI-driven approach could theoretically incorporate filter-like constraints within natural language inputs, we retained traditional filters to maintain a consistent user experience and facilitate direct comparison. This design empowers participants to evaluate both methods simultaneously, offering insights into how each aligns with their search preferences and behaviors.

### 1.1 Implementation of the Interface

Initially, we aimed to leverage a local large language model for dynamic SQL query generation, drawn by the advantages of cost-free access, robust documentation, and adaptability. However, hardware constraints quickly surfaced. Testing models like Meta Llama-3.3-70B-Instruct [**?** ] and Microsoft Phi-4 [**?** ] from Hugging Face on a system with 8GB RAM proved unfeasible due to insufficient memory and suboptimal performance, even with quantized versions (e.g., GGUF models). These limitations prompted a shift to an API-based solution. We selected the Claude API, which has demonstrated superior performance in complex tasks like programming and translation compared to alternatives like ChatGPT [**?** ].

For the AI-driven approach, we crafted a prompt that includes the database schema, a comprehensive list of available features (e.g., "Hündin" or "Rüde" for gender), and a base query to guide the generation process. This base query, shown in Fig. **??**, provides a standardized structure that ensures consistent table relationships and attribute retrieval, which the AI refines based on user input. This setup enables the system to interpret both vague and detailed queries accurately, mapping them to appropriate SQL statements. The static template-based method employs a streamlined design, tokenizing user input and applying an OR-condition across dog breeds and descriptions (Fig. **??**). Our static template was designed to provide a fair basis for comparison by reflecting typical keyword-matching techniques used in many e-commerce systems.

### 1.2 User Study

Our study involved 13 participants—12 male and one female—predominantly computer science students pursuing Bachelors or Masters degrees, many with prior exposure to artificial intelligence through coursework or personal interest. This convenience sample, drawn from our academic network, is small and relatively homogeneous but sufficient for an exploratory investigation. Participants were briefed on the study's goal: to compare the AI-driven search system

**Base-Query**

```
SELECT h.*, GROUP_CONCAT(F.Name)
FROM Hund h
INNER JOIN
main.Feature_Hund FH on h.id = FH.Hund_id
INNER JOIN main.Feature F on F.id = FH.Feature_id
```

Fig. 1. Base query utilized in our study, dynamically extended by the Claude API. The GROUP_CONCAT function aggregates all associated features into a single entry per dog, while the predefined structure ensures a consistent result order, such as the sequence of attributes, essential for our search implementation.

**Static Template Implementation**

```
SELECT h.*, GROUP_CONCAT(F.Name)
FROM Hund h
INNER JOIN main.Feature_Hund FH on h.id = FH.Hund_id
INNER JOIN main.Feature F on F.id = FH.Feature_id
WHERE (h.description LIKE '%Labrador%' AND h.breed LIKE '%Labrador%')
   OR (h.description LIKE '%freundlich%' AND h.breed LIKE '%freundlich%')
```

Fig. 2. Static query template implementation exemplified with the input 'Labrador freundlich'. For each token ('Labrador' and 'freundlich', meaning 'friendly' in German), the template generates LIKE conditions to match both the h.breed and h.description attributes, linked by OR, extending the base query for keyword-based search.

against a conventional template-based approach and assess its potential to enhance product search accuracy and user satisfaction. We hypothesized that the AI-generated queries would outperform the static method, particularly for vague, descriptive, or multi-criteria searches, while the static approach would remain competitive for simple, attribute-based queries.

The study unfolded in three phases. First, participants answered general questions about their past search experiences, focusing on challenges encountered and their impact on shopping behavior. In the second phase, they interacted with the interface, completing six predefined search tasks designed to explore varying levels of complexity:

- Find the Labrador Retriever
- Find the most expensive Labrador Retriever
- Find the most expensive Labrador Retriever (without price filter)
- Find all dogs except Australian Shepherd
- Find Chihuahuas that are eager to work
- Find all Rottweilers that are senior-friendly or German Shepherds younger than 12 months

These tasks progressed from basic attribute searches to complex queries unachievable with filters alone, such as those requiring logical operators like "OR" across distinct attributes. Participants could freely use natural language or static filters (unless specified), with the option to refine queries if results fell short. Following this guided exploration, they were encouraged to devise their own complex searches. The final phase involved a detailed questionnaire with a 5-point Likert scale assessing accuracy, ease of use, and efficiency, complemented by open-ended questions for qualitative insights (please refer to Appendix **??** for an overview). This mixed-methods approach illuminated user preferences,

system strengths, and areas for refinement, while also probing whether AI-driven search could supplant traditional filtering mechanisms.

**Temporary page!**

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.