

# **Input and Output for ChatGPT**

## **Lindsey Chenault**

### **Requirement 3)**

**My input:**

“You are a database application developer.

Follow these steps to answer the user queries:

Step 1: Parse the user query to determine the type of the query and expected output/result of the query.

Step 2: Determine the type of the required database system by performing the following steps:

- Step 2.1: If the query asks for results in the form of numerical values, calculations, aggregations, or structured data, then use \*\*PostgreSQL\*\*.
- Step 2.2: If the query asks for exact/fuzzy search of strings in documents, then use \*\*Elasticsearch\*\*.
- Step 2.3: If the query asks for similarity/semantic/hybrid search, then use \*\*Milvus\*\*.

Step 3: Use the database system you identified in Step 2 and match it to the correct query.

Now, based on the steps above, identify the best database system for the following queries:

1. I have an Amazon product reviews dataset. Which database system should I use for this type of query:  
How many products have an average review rating below 2.5?
2. I have an Amazon product reviews dataset. Which database system should I use for this type of query:  
How many reviewers have written at least 3 reviews in the last year?
3. I have an Amazon product reviews dataset. Which database system should I use for this type of query:  
What is the average number of reviews with a 4.5-star rating per product category?
4. I have an Amazon product reviews dataset. Which database system should I use for this type of query:  
Find reviews that mention “easy” but do not include “again.”
5. I have an Amazon product reviews dataset. Which database system should I use for this type of query:  
Retrieve reviews where users misspelled “Refrigerator Parts” and rated with 3.5 stars or below.
6. I have an Amazon product reviews dataset. Which database system should I use for this type of query:  
Get a list of reviews where the reviewer mentioned “bad quality” within five words of “expensive.”
7. I have an Amazon product reviews dataset. Which database system should I use for this type of query:  
Find reviews most similar to “The disposal worked and was installed in less time than expected.”
8. I have an Amazon product reviews dataset. Which database system should I use for this type of query:  
Identify reviews that are similar to “After a couple of years, the unit started freezing up,” but only for products with more than 50 reviews.
9. I have an Amazon product reviews dataset. Which database system should I use for this type of query:

Retrieve reviews that are semantically related to “The oven heats unevenly” and were written in 2020 or later.

Please return your response in a table format with two columns:

1. Query
2. Recommended Database (PostgreSQL, Elasticsearch, or Milvus)”

**Output:**

Here is the recommended database system for each query based on the steps you provided:

Query	Recommended Database
How many products have an average review rating below 2.5?	PostgreSQL
How many reviewers have written at least 3 reviews in the last year?	PostgreSQL
What is the average number of reviews with a 4.5-star rating per product category?	PostgreSQL
Find reviews that mention “easy” but do not include “again.”	Elasticsearch
Retrieve reviews where users misspelled “Refrigerator Parts” and rated with 3.5 stars or below.	Elasticsearch
Get a list of reviews where the reviewer mentioned “bad quality” within five words of “expensive.”	Elasticsearch
Find reviews most similar to “The disposal worked and was installed in less time than expected.”	Milvus
Identify reviews that are similar to “After a couple of years, the unit started freezing up,” but only for products with more than 50 reviews.	Milvus
Retrieve reviews that are semantically related to “The oven heats unevenly” and were written in 2020 or later.	Milvus

- PostgreSQL is best for structured data queries involving numerical values, aggregations, and filtering.

Ask anything



Search



Reason



ChatGPT can make mistakes. Check important info.

- **PostgreSQL** is best for structured data queries involving numerical values, aggregations, and filtering.
- **Elasticsearch** is best for text search, including keyword-based queries, fuzzy matching, and phrase proximity.
- **Milvus** is best for similarity and semantic search.

Would you like me to refine the selection further based on additional constraints?

#### **Requirement 4)**

**My input:**

- I have an Amazon product reviews dataset. Use the PostgreSQL database system and create code for the query: How many products have an average review rating below 2.5?
- I have an Amazon product reviews dataset. Use the PostgreSQL database system and create code for the query: How many reviewers have written at least 3 reviews in the last year?
- I have an Amazon product reviews dataset. Use the PostgreSQL database system and create code for the query: What is the average number of reviews with a 4.5-star rating per product category?
- I have an Amazon product reviews dataset. Use the Elasticsearch database system and create code for the query: Find reviews that mention “easy” but do not include “again.”
- I have an Amazon product reviews dataset. Use the Elasticsearch database system and create code for the query: Retrieve reviews where users misspelled “Refrigerator Parts” and rated with 3.5 stars or below.
- I have an Amazon product reviews dataset. Use the Elasticsearch database system and create code for the query: Get a list of reviews where the reviewer mentioned “bad quality” within five words of “expensive.”
- I have an Amazon product reviews dataset. Use the Milvus database system and create code for the query: Find reviews most similar to “The disposal worked and was installed in less time than expected.”
- I have an Amazon product reviews dataset. Use the Milvus database system and create code for the query: Identify reviews that are similar to “After a couple of years, the unit started freezing up,” but only for products with more than 50 reviews.
- I have an Amazon product reviews dataset. Use the Milvus database system and create code for the query: Retrieve reviews that are semantically related to “The oven heats unevenly” and were written in 2020 or later.

**Output:**

1. How many products have an average review rating below 2.5?

```
sql
```

```
SELECT COUNT(product_id)
FROM (
    SELECT product_id, AVG(rating) AS avg_rating
    FROM reviews
    GROUP BY product_id
    HAVING AVG(rating) < 2.5
) subquery;
```

Copy  Edit

2. How many reviewers have written at least 3 reviews in the last year?

```
sql
```

```
SELECT COUNT(review_id)
FROM (
    SELECT review_id, COUNT(*) AS review_count
    FROM reviews
    WHERE review_date >= NOW() - INTERVAL '1 year'
    GROUP BY review_id
    HAVING COUNT(*) >= 3
) subquery;
```

Copy  Edit

Ask anything



Search



Reason



3. What is the average number of reviews with a 4.5-star rating per product category?

```
sql
SELECT category, AVG(review_count) AS avg_reviews
FROM (
    SELECT category, product_id, COUNT(*) AS review_count
    FROM reviews r
    JOIN products p ON r.product_id = p.product_id
    WHERE rating = 4.5
    GROUP BY category, product_id
) subquery
GROUP BY category;
```

## Elasticsearch Queries

4. Find reviews that mention "easy" but do not include "again."

```
json
{
  "query": {
    "bool": {
      "must": {
        "match": {
          "review_text": "easy"
        }
      },
      "must_not": {
        "match": {
          "review_text": "again"
        }
      }
    }
  }
}
```

5. Retrieve reviews where users misspelled "Refrigerator Parts" and rated with 3.5 stars or below.

```
json
Copy Edit

{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "review_text": {
              "query": "Refrigerator Parts",
              "fuzziness": "AUTO"
            }
          }
        },
        {
          "range": {
            "rating": {
              "lte": 3.5
            }
          }
        }
      ]
    }
  }
}
```

6. Get a list of reviews where the reviewer mentioned "bad quality" within five words of "expensive."

```
json
Copy Edit

{
  "query": {
    "span_near": {
      "clauses": [
        { "span_term": { "review_text": "bad quality" } },
        { "span_term": { "review_text": "expensive" } }
      ],
      "slop": 5,
      "in_order": false
    }
  }
}
```

## Milvus Queries

7. Find reviews most similar to “The disposal worked and was installed in less time than expected.”

python

 Copy  Edit

```
from pymilvus import Collection
import numpy as np

collection = Collection("reviews")
query_vector = embed("The disposal worked and was installed in less time than expected")

search_params = {"metric_type": "IP", "params": {"nprobe": 10}}

results = collection.search(
    data=[query_vector],
    anns_field="review_embedding",
    param=search_params,
    limit=10,
    output_fields=["review_text"]
)

for result in results[0]:
    print(result.entity.get("review_text"))
```

Ask anything

8. Identify reviews that are similar to "After a couple of years, the unit started freezing up," but only for products with more than 50 reviews.

```
python

query_vector = embed("After a couple of years, the unit started freezing up.")

search_results = collection.search(
    data=[query_vector],
    anns_field="review_embedding",
    param={"metric_type": "IP", "params": {"nprobe": 10}},
    limit=10,
    output_fields=["review_text", "product_id"]
)

filtered_results = []
for result in search_results[0]:
    product_id = result.entity.get("product_id")

    # Check if the product has more than 50 reviews
    review_count = collection.query(
        expr=f"product_id == '{product_id}'",
        output_fields=["count(*)"]
    )[0]["count(*)"]

    if review_count > 50:
```

Ask anything

```
    if review_count < 50:  
        filtered_results.append(result.entity.get("review_text"))  
  
print(filtered_results)
```

 Copy  Edit

9. Retrieve reviews that are semantically related to "The oven heats unevenly" and were written in 2020 or later.

```
python  
  
query_vector = embed("The oven heats unevenly.")  
  
search_results = collection.search(  
    data=[query_vector],  
    anns_field="review_embedding",  
    param={"metric_type": "IP", "params": {"nprobe": 10}},  
    limit=10,  
    output_fields=["review_text", "review_date"]  
)  
  
filtered_reviews = [  
    result.entity.get("review_text")  
    for result in search_results[0]  
    if result.entity.get("review_date") >= "2020-01-01"  
]
```

 Copy  Edit

Ask anything

```
search_results = collection.search(
    data=[query_vector],
    anns_field="review_embedding",
    param={"metric_type": "IP", "params": {"nprobe": 10}},
    limit=10,
    output_fields=["review_text", "review_date"]
)

filtered_reviews = [
    result.entity.get("review_text")
    for result in search_results[0]
    if result.entity.get("review_date") >= "2020-01-01"
]

print(filtered_reviews)
```

Copy Edit

This covers the required queries for PostgreSQL, Elasticsearch, and Milvus. Let me know if you need modifications or explanations! 🚀



Ask anything



Search

Reason



ChatGPT can make mistakes. Check important info.