Pokémon Market Natural Language Query System

Zixi Wang

4/22/2025 DSCI 551

This system queries the MongoDB database through natural language (NLQ)

Collections in Pokémon Market Database:

- 1. card_informations: Pokémon card details
- 2. card_price: Market prices of cards
- 3. card_sets: Set information
- 4. deck_cards: Cards in each deck
- 5. pokemon_decks: Deck meta info
- 6. pokemon_game: Game release info
- 7. pokemon_movies: Movie release info
- 8. pokemon_information: Pokémon base stats

Collection Details

card_informations
- id
- name
- subtypes
- hp
- types
- attacks
- weaknesses
- artist
- nationalPokedexNumbers
- rarity
- set_id

card_price
- name
- set_id
- price
- title

card_sets
- id
- name
- series
- releaseDate
- legalities
- total
- generation

deck_cards
- deck_id
- id
- name
- rarity
- count

pokemon_decks
- id
- name
- types

pokemon_game
- game_id
- title_japanese
- title_english
- release_date_japan
- generation
- generation_short
- platform
- starter_pokemon
- legendary_pokemon
- mythical_pokemon
- main_character

Collection Details

pokemon_movies

- movie_id
- japanese_title
- english_title
- release_date_japan
- generation
- generation_short
- featured_pokemon

pokemon_information

- id
- name
- type
- base
- evolutions

Important Relationships:

card_informations.name = card_price.name

card_price.set_id = card_sets.id

card_informations.set_id = card_sets.id

card_price is the relational table for card_informations and card_sets

deck_cards.deck_id = pokemon_decks.id

deck_cards.name = card_informations.name

deck_cards is the relational table for card_informations and pokemon_decks

pokemon_game.generation = pokemon_movies.generation = card_sets.generation

pokemon_game more vs more pokemon_movies

pokemon_movies more vs more card_sets

pokemon_information.name = card_informations.name = deck_cards.name

pokemon_information 1 vs more card_informations

Workflow

User input → app.py (controls frontend interaction)

app.py → nlp_transfer.py (calls OpenAI to generate MongoDB query)

nlp_transfer.py → returns MongoDB query

app.py → mongodb.py (executes query in MongoDB)

mongodb.py → returns results to app.py

app.py → outputs formatted results to user

app.py: Handles user input, manages control flow, and outputs results.

nlp_transfer.py: Converts natural language into structured MongoDB queries using OpenAI API.

mongodb.py: Connects to the MongoDB database and executes operations such as find, aggregate, insert, update, and delete.

data_upload.ipynb: A separate Jupyter Notebook (data_upload.ipynb) was used to clean, transform, and upload data into the MongoDB database.

Test Query

1. [find + projection] Find the name and HP of all Pokémon cards with the type "Fire"
2. [aggregate + $unwind + $group] What is the average HP for each Pokémon type
3. [aggregate + $match + $group] Among rare cards only, show the average HP by type
4. [aggregate + $group + $match] Show types with average HP above 100
5. [find + skip + limit] Skip the first 10 cards and show the next 5 cards with type Grass
6. [aggregate + $lookup + $project] Show the price and info of all Charizard cards
7. [aggregate + $sort + $limit] Show the top 5 most expensive Pokémon cards
8. [aggregate + $group + count + match] For each generation, count how many movies were released
9. [aggregate + $lookup + $match] Give me the price for Grass type Pokémon
10. [aggregate + $match + $count] Count me how many VMAX cards

Test Query

11. **[insertOne] Insert a new Pokémon card named "Testchu" with type Electric and HP 60**
12. **[insertMany] Insert two new cards: "Testchu2" and "Testchu3", both Electric type**
13. **[updateOne] Update the HP of "Testchu" to 100**
14. **[deleteOne] Delete the card named "Testchu2"**

15. **give me the time latest three results for card_information collection**

Thank You