Data Dogs Final Project

By: Bridget Cullen and Lindsay Lebowitz

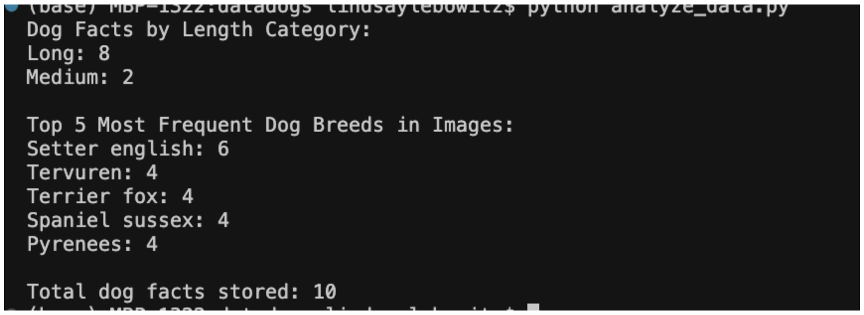
A. The goal of this project was to explore data related to dog facts and breeds using APIs. We chose to work with the following two APIs:
   - **Dog Facts API:** To retrieve interesting and random facts about dogs.
   - **Dog CEO API:** To fetch images of different dog breeds and a list of all available breeds.

   This would allow us to work with a mix of text and image data, and store everything in a single SQLite database for analysis and visualization.

B. We integrated both the Dog Facts API and the Dog CEO API that we originally planned to use:
   - **Dog Facts API** – We gathered dog facts, calculated the length of each fact, and categorized them as short, medium, or long. This data was stored in a DogFacts table, with each fact linked to a FactLengthCategory table using a foreign key.
   - **Dog CEO API** – We gathered 25 random dog images per run and extracted the breed name from each image URL. This data was stored in a DogBreeds table, which includes the breed name and the image URL.

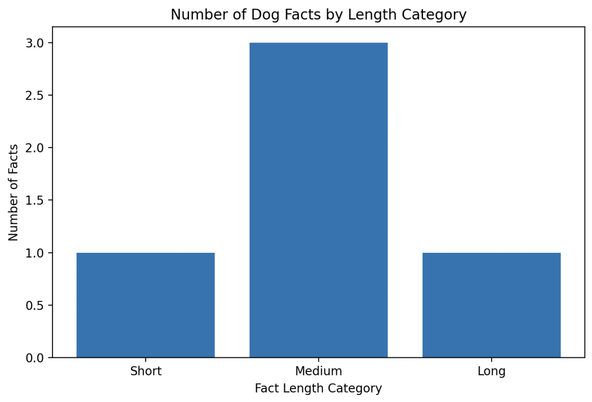
C. Throughout the project, we faced a few challenges. Accessing the Dog Facts API occasionally returned errors or slow responses, which made gathering consistent data more difficult. We also ran into issues when transferring code between each other, especially making sure we were both working with the most updated version. Using GitHub together was initially tricky, as we had to learn how to use git pull, and resolve merging conflicts when working on the same files. We were able to communicate effectively and work through the problems to complete the project successfully.

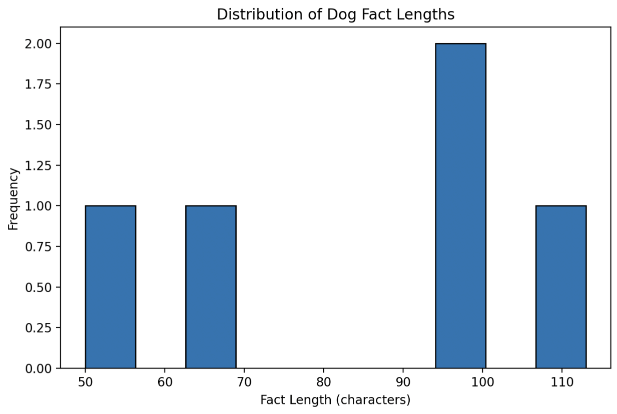
D. We performed calculations using data from both tables in our database. From the DogFacts table, we calculated how many facts fell into each category: short, medium, or long. This allowed us to understand the distribution of fact lengths and supported a bar chart visualization. From the DogBreeds table, we counted how many times each breed appeared across the randomly selected dog images. This helped us identify the most frequently appearing breeds. These calculations were done using Python, and we included a screenshot of the results from our terminal to show our outcome.

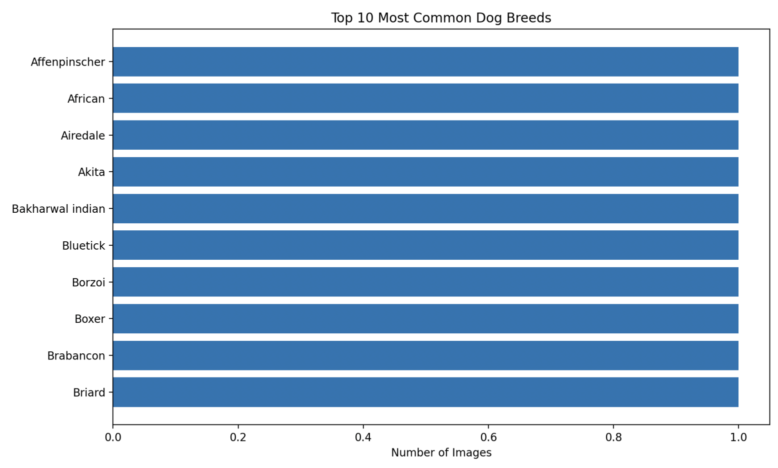


```
(base) MBP-1322:datadogs lindsaylebowitz$ python analyze_data.py
Dog Facts by Length Category:
Long: 8
Medium: 2

Top 5 Most Frequent Dog Breeds in Images:
Setter english: 6
Tervuren: 4
Terrier fox: 4
Spaniel sussex: 4
Pyrenees: 4

Total dog facts stored: 10
```

E.







F. We ran dog_facts_api.py multiple times to gather at least 100 dog facts, each of which was categorized by length and stored in a SQLite database (final_project.db). We ran dog_ceo_api.py several times also to collect 100+ random dog images and extract the breed name from each one, which was also stored in the database. To analyze our data, we ran analyze_data.py, which prints out summary statistics in the terminal, including fact counts by category, the top 5 most common dog breeds, and the total number of stored facts. To generate and view visualizations, we ran process_data.py, which creates three charts: fact_length_bar_chart.png, fact_length_histogram.png, and top_breeds_bar_chart.png. Lastly, to view the database tables, we used DB Browser for SQLite to open final_project.db, navigate to the "Browse Data" tab, and inspect the DogFacts, FactLengthCategory, and DogBreeds tables.

G.

dog_facts_api.py:

init_db()

- Purpose: Initializes the database by creating the DogFacts and FactLengthCategory tables if they don't exist.
- Input: None
- Output: None

categorize_length(length)

- Purpose: Categorizes the length of a dog fact as "Short", "Medium", or "Long".
- Input: length (int) the number of characters in a fact
- Output: A string category: "Short", "Medium", or "Long"

fetch_and_store_facts()

- Purpose: Fetches up to 25 dog facts from the API, categorizes them by length, and stores them in the DogFactstable along with their length and category ID.
- Input: None
- Output: None (modifies database directly)

dog_ceo_api.py:

reset_db()

- Purpose: Deletes the DogImages and DogBreeds tables if they already exist. This resets the database.
- Input: None
- Output: None

init_db()

- Purpose: Creates two tables. DogBreeds which stores dog breed names and DogImages which stores image URLs linked to dog breeds using a foreign key.
- Input: None
- Output: None (modifies database directly)

parse_breed_from_url(url)

- Purpose: Extracts the breed name from a Dog CEO image URL.
- Input: url (str) – a URL pointing to a dog image
- Output: A string containing the formatted breed name

fetch_and_store_breeds()

- Purpose: Fetches 25 random dog images and stores the image URL and breed name in the DogBreeds table.
- Input: None
- Output: None

process_data.py:

calculate_and_write_stats()

- Purpose: Calculates the average fact length, counts facts by category, counts total breeds, and writes results to a .txt file.
- Input: None
- Output: Returns category_counts – a list of tuples with category names and their respective fact counts

create_bar_chart(category_counts)

- **Purpose:** Creates a bar chart showing the number of facts per length category.
- **Input:** category_counts (list of tuples)
- **Output:** Saves the chart as fact_length_bar_chart.png and displays it

create_histogram()

- **Purpose:** Creates a histogram of dog fact lengths.
- **Input:** None
- **Output:** Saves the chart as fact_length_histogram.png and displays it

create_breed_bar_chart()

- **Purpose:** Creates a horizontal bar chart of the top 10 most common dog breeds in the database.
- **Input:** None
- **Output:** Saves the chart as top_breeds_bar_chart.png and displays it

H.

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
4/7/25	Couldn't figure out how to connect to the Dog Facts API.	ChatGPT	Yes. Reviewed documentation and used requests.get()
4/7/25	Received data from API but didn’t know how to store it in the	ChatGPT	Yes. Created INSERT INTO statements and tested functionality.

	SQLite database		
4/8/25	Unsure of how to categorize fact lengths into short, medium, long.	ChatGPT	Yes. Added if-else statements with character count thresholds.
4/10/25	Breed bar chart was not displaying.	ChatGPT	Yes. Had to fix a typo in the table name.
4/10/25	Error joining two tables in SQL query.	ChatGPT	Yes. Corrected join syntax and checked for matching keys
4/13/25	Error in calculations	ChatGPT	Yes. Found a logic error and added print statements for debugging
4/17/25 - after grading session	Had repeats in the dog breed SQLite table	Grading Session	Yes. Identified and removed duplicate entries by checking for existing records before inserting new ones into the dog breed table. Added a separate table for image URLs to prevent redundancy.