fire_index_mapping_with_fire_boundaries_refined.py

Explanation:

SQLite Data Storage:

- **Database Connection and Table Creation:** It establishes a connection to an SQLite database (fire_weather.db) and creates a table named fire_data if one doesn't already exist. This table is designed to store the calculated fire indices along with location (latitude, longitude), date, temperature, relative humidity, wind speed, and precipitation.
- **Data Insertion:** Inside the loop that iterates through the la_county_grid, after fetching weather data and calculating the fire indices (FFDI, FWI, mFFWI), an SQL INSERT statement is used to store this data into the fire_data table. conn.commit() ensures that the data is persistently written to the database.

SQLite Data Retrieval:

- **Database Query:** Before creating the map visualization, an SQL SELECT statement is executed to retrieve the necessary data (latitude, longitude, FFDI, FWI, mFFWI) from the fire_data table for the current date (today). This query fetches all rows where the date column matches the current date.
- **Populating grid_data:** The results of the database query (grid_data_from_db) are then used to populate the grid_data dictionary. This dictionary is structured to be compatible with the create_fire_index_map function, which requires a specific format to generate the visualization. Each entry in grid_data corresponds to a location (latitude, longitude) and holds the associated fire index values. This dictionary is then used to create the visualization.

In summary, the code first calculates and stores fire indices for different locations in the database. Then, it retrieves this stored data to create the map visualization, demonstrating a complete cycle of data storage and retrieval using SQLite. This approach ensures that the visualization is generated from the data stored in the database, not from the intermediate calculations within the loop.