Example Structure for Redis Blog Database

1. **Design the Data Structure**: Each blog post will have a title, text, author, and time/date. We'll use a unique identifier (like a post ID) for each blog post.

2. **Use Redis Hashes**: Redis hashes are ideal for storing objects with multiple fields. We'll create a hash for each blog post.

3. **Commands to Create Blog Posts**: For each blog post, you'll use a command like this:

   ```
   HMSET blogpost:<id> title "Post Title" text "Post text here" author "Author Name" date "YYYY-MM-DD HH:MM"
   ```

   Replace `<id>` with a unique identifier for each post.

4. **Retrieve a Blog Post**: To get a blog post, use:

   ```
   HGETALL blogpost:<id>
   ```

5. **List All Blog Posts**: If you want to list all blog posts, you'll need to maintain a list or set of all blog post IDs. When a new post is created, add its ID to this list/set.

   - To create the list/set: `SADD blogposts <id>`
   - To retrieve all posts: `SMEMBERS blogposts`

6. **Commands for Dataset Creation**: Here's an example set of commands to create a dataset of 3 blog posts:

   ```
   HMSET blogpost:1 title "First Post" text "This is the first blog post" author "John Doe" date "2023-01-01 10:00"
   HMSET blogpost:2 title "Second Post" text "This is the second blog post" author "Jane Doe" date "2023-01-02 11:00"
   HMSET blogpost:3 title "Third Post" text "This is the third blog post" author "Jim Doe" date "2023-01-03 12:00"

   SADD blogposts 1 2 3
   ```

7. **Retrieving the Dataset**: To retrieve the entire dataset, you would iterate over the IDs in the `blogposts` set and fetch each post using `HGETALL`.

Remember, this is a basic example. Depending on your requirements, you might need to implement additional features like pagination, indexing for faster search, or even use Redis' JSON module if you're dealing with more complex data structures.