Optimization Analysis of the API and SQL Queries

1. API Optimization

The API demonstrates several optimization techniques that enhance performance, security, and maintainability:

- **Singleton Pattern**:
  The API class uses a singleton pattern (getInstance()), ensuring only one database connection is maintained per request, reducing overhead.
- **Input Validation**:
  All incoming JSON data is validated early (json_decode error checking), preventing malformed requests from proceeding further.
- **Modular Error Handling**:
  Centralized error handling via sendError() and sendSuccess() ensures consistent responses and proper HTTP status codes.
- **Parameterized Queries**:
  SQL queries use prepared statements with bound parameters, preventing SQL injection and improving query reusability.
- **Efficient Data Fetching**:
  Methods
  like handleGetAllProducts() and handleGetDistinct() use LIMIT clauses to restrict result sizes, reducing memory usage and network load.
- **CORS Headers**:
  Proper CORS headers (Access-Control-Allow-Origin, etc.) are set to manage cross-origin requests securely.
- **Authentication Checks**:
  API endpoints validate the apikey parameter before processing requests, ensuring only authorized users access data.

2. SQL Query Optimization

The SQL queries are optimized for performance and scalability:

- **Indexing Implicit Use**:
  Queries filter on primary keys (e.g., ProductID, API_Key), which are typically indexed, speeding up lookups.
- **JOIN Optimization**:
  In handleGetAllProducts(), joins are minimized by only linking necessary tables (products, productratings), reducing computational overhead.
- **Aggregation Efficiency**:
  The handleGetAllProducts() query

calculates averageRating and ratingCount in a single pass using AVG() and COUNT(), avoiding multiple subqueries.

- **Dynamic Query Building**:
  Methods like handleGetDistinct() construct queries dynamically based on input parameters, ensuring only relevant tables and columns are queried.
- **Bounded Limits**:
  Queries enforce limits (e.g., LIMIT :limit with bounds like 1-100) to prevent excessive data retrieval.
- **Conditional Filtering**:
  The handleGetDistinct() method supports optional filters (e.g., minRating), dynamically adjusting the WHERE clause to avoid full-table scans.
- **Batch Operations**:
  For wishlist and rating updates, operations like INSERT/UPDATE are batched, reducing round trips to the database.