AEROASPIRE INTERN ASSESSMENT-03
KARTHIK Tl

1. How do you design efficient queries for filtering? What is an index and when do you use it?

Efficient query design for filtering focuses on reducing the amount of data the database scans. Using selective WHERE clauses with equality or range conditions helps narrow down results quickly. Avoid using SELECT * and instead select only necessary columns. Indexes are key to efficiency—they are data structures that act like a lookup table to speed up query searches by allowing the database to quickly locate rows matching filter conditions without scanning the entire table. You should use indexes on columns frequently used in WHERE, JOIN, or ORDER BY clauses, especially in large tables. However, overusing indexes can slow writes and consume storage, so use them judiciously.

2. How does pagination work (offset/limit etc.)?

Pagination controls how many records a query returns to avoid overwhelming the client or server. The most common method uses LIMIT to define how many rows to fetch, and OFFSET to skip a number of rows before starting to return results. For example, LIMIT 10 OFFSET 20 fetches 10 records starting from the 21st row. This allows clients to request data page by page. However, high OFFSET values can degrade performance because the database still scans those skipped rows. Alternatives like cursor-based pagination can be used for better performance in large datasets.

3. Flow of building endpoints, receiving query params, applying them in SQL/ORM, returning results

When building endpoints that support filtering and pagination:

- Receive query parameters from the client, such as filters, limits, and offsets.
- Validate and sanitize these parameters.
- Use the ORM or SQL queries to apply the filters (WHERE clauses) and pagination (LIMIT, OFFSET).
- Execute the query against the database.
- Return the result set to the client, usually as JSON.
- Optionally include metadata like total count or paging info for frontend navigation.

This approach keeps APIs flexible, efficient, and scalable by allowing clients to control data shape and size dynamically.