Optomising Query Performance

Optimizing database queries is essential for improving performance, reducing load times, and enhancing scalability. Here are some key strategies:

### **1. Indexing**

* **Primary & Unique Indexes**: Ensure that columns used in WHERE, JOIN, and ORDER BY clauses have proper indexes.
* **Composite Indexes**: Use multi-column indexes for queries filtering by multiple columns.
* **Covering Indexes**: Include all queried columns in an index to reduce lookups.

### **2. Optimize Query Structure**

* **Avoid SELECT \***: Select only necessary columns to reduce data transfer.
* **Use WHERE Clauses Efficiently**: Filter results early to limit data processing.
* **Optimize JOINs**: Use indexed columns for joining large tables

### **3. Optimize Data Storage**

* **Normalize Data**: Reduce redundancy and improve consistency.
* **Denormalize When Necessary**: Precompute complex joins in separate tables for faster retrieval.
* **Partitioning & Sharding**: Distribute data across multiple tables or servers.

### **4. Use Caching**

* **Query Caching**: Store results of frequent queries in memory.
* **Application-Level Caching**: Use tools like Redis or Memcached for frequently accessed data.

### **5. Analyze Execution Plans**

* Use EXPLAIN (MySQL, PostgreSQL) or EXPLAIN ANALYZE to understand query execution.
* Identify slow operations like full table scans or inefficient joins.