**Section 1: SQL Basics (Q1–Q20)**

1. **What is SQL?**
Structured Query Language – used to access, manage, and modify data in databases.
2. **Difference between SQL and MySQL?**
SQL is a language; MySQL is a database management system that uses SQL.
3. **What is a database?**
An organized collection of data.
4. **What is DBMS?**
Database Management System – software to create, store, and manage databases.
5. **What is RDBMS?**
Relational DBMS – stores data in tables with rows and columns.
6. **Main types of SQL statements?**
DDL, DML, DCL, TCL.
7. **What is DDL?**
Data Definition Language – defines and alters database structure (CREATE, ALTER, DROP).
8. **What is DML?**
Data Manipulation Language – modifies data (INSERT, UPDATE, DELETE).
9. **What is DCL?**
Data Control Language – manages permissions (GRANT, REVOKE).
10. **What is TCL?**
Transaction Control Language – manages transactions (COMMIT, ROLLBACK, SAVEPOINT).
11. **What is a table in SQL?**
A structure with rows and columns to store data.
12. **What are rows and columns?**
Row = record; Column = field.
13. **What is a primary key?**
A column with unique and non-null values.
14. **What is a foreign key?**
A column referencing the primary key in another table.
15. **What is NULL in SQL?**
Represents missing or unknown value (not the same as empty).
16. **Purpose of SELECT statement?**
Retrieve data from tables.
17. **What are aliases in SQL?**
Temporary names for columns or tables.
18. **Purpose of DISTINCT keyword?**
Returns unique rows by removing duplicates.
19. **What is ORDER BY clause?**
Sorts results in ascending or descending order.
20. **What is WHERE clause?**
Filters rows based on a condition.

**Section 2: Filtering & Operators (Q21–Q40)**

1. **Difference between AND and OR?**
AND – both conditions must be true; OR – at least one must be true.
2. **What is BETWEEN?**
Filters values within a range.
3. **What is IN?**
Matches values from a list.
4. **What is LIKE?**
Matches a pattern.
5. **Difference between % and \_ wildcards?**
% – multiple characters; \_ – single character.
6. **IS NULL vs IS NOT NULL?**
Checks if value is null or not null.
7. **Comparison operators in SQL?**
=, !=, <>, >, <, >=, <=.
8. **Logical operators in SQL?**
AND, OR, NOT.
9. **Arithmetic operators in SQL?**
+, -, \*, /.
10. **Purpose of NOT keyword?**
Negates a condition.
11. **What is ALL keyword?**
Compares a value to all values in a subquery.
12. **What is ANY keyword?**
Compares a value to any value in a subquery.
13. **What is EXISTS?**
Checks if subquery returns any rows.
14. **Difference between UNION and UNION ALL?**
UNION removes duplicates; UNION ALL keeps duplicates.
15. **What is INTERSECT?**
Returns common rows from both queries.
16. **What is MINUS (or EXCEPT)?**
Returns rows from first query not in the second.
17. **What is CASE statement?**
Implements conditional logic in SQL.
18. **What is COALESCE?**
Returns the first non-null value.
19. **What is NULLIF?**
Returns NULL if two expressions are equal.
20. **Difference between CAST and CONVERT?**
Both change data type; syntax differs by database.

**Section 3: Joins (Q41–Q60)**

1. **What is a join?**
Combines related data from multiple tables.
2. **What is INNER JOIN?**
Returns only matching rows from both tables.
3. **What is LEFT JOIN?**
All rows from left table, matched rows from right.
4. **What is RIGHT JOIN?**
All rows from right table, matched rows from left.
5. **What is FULL OUTER JOIN?**
All rows from both tables, matching where possible.
6. **What is CROSS JOIN?**
Cartesian product of two tables.
7. **What is SELF JOIN?**
Joining a table with itself.
8. **What is NATURAL JOIN?**
Joins on columns with the same names.
9. **What is EQUI JOIN?**
Join using equality condition.
10. **What is NON-EQUI JOIN?**
Join using non-equality condition.
11. **What is Theta Join?**
Join using any comparison operator.
12. **What is Hash Join?**
Uses hash tables for faster joins.
13. **What is Merge Join?**
Joins sorted tables by merging.
14. **What is Anti Join?**
Returns rows in one table with no match in another.
15. **What is Semi Join?**
Returns rows that match, without duplicates.
16. **Join vs Subquery?**
Join merges tables; subquery is a nested query.
17. **Can we join more than two tables?**
Yes, multiple joins are allowed.
18. **Can we join without ON clause?**
Yes, in CROSS JOIN.
19. **Which join returns maximum rows?**
CROSS JOIN.
20. **Which join returns minimum rows?**
INNER JOIN.

**Section 4: Aggregate & Scalar Functions (Q61–Q80)**

1. **What is an aggregate function?**
A function that returns a single value from multiple rows (e.g., SUM, AVG).
2. **What is a scalar function?**
Returns a single value for each row (e.g., UPPER, LOWER).
3. **What does COUNT() do?**
Counts rows.
4. **What does SUM() do?**
Adds values in a numeric column.
5. **What does AVG() do?**
Returns the average of numeric values.
6. **What does MIN() do?**
Returns the smallest value in a column.
7. **What does MAX() do?**
Returns the largest value in a column.
8. **What does ROUND() do?**
Rounds a numeric value to a specified precision.
9. **Difference between CEIL() and FLOOR()?**
CEIL returns the next integer; FLOOR returns the previous integer.
10. **What does ABS() do?**
Returns the absolute value.
11. **What does POWER() do?**
Returns a number raised to a power.
12. **What does MOD() do?**
Returns the remainder after division.
13. **What does UPPER() do?**
Converts string to uppercase.
14. **What does LOWER() do?**
Converts string to lowercase.
15. **What does INITCAP() do?**
Capitalizes the first letter of each word.
16. **What does LTRIM() do?**
Removes leading characters from a string.
17. **What does RTRIM() do?**
Removes trailing characters from a string.
18. **What does TRIM() do?**
Removes both leading and trailing characters.
19. **What does SUBSTR() do?**
Extracts part of a string.
20. **What does REPLACE() do?**
Replaces part of a string with another value.

**Section 5: Constraints (Q81–Q100)**

1. **What is a constraint?**
A rule that enforces data integrity.
2. **Types of constraints?**
PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK, DEFAULT.
3. **What is PRIMARY KEY constraint?**
Unique and not null.
4. **What is FOREIGN KEY constraint?**
References the primary key of another table.
5. **What is UNIQUE constraint?**
Ensures all values are different.
6. **What is NOT NULL constraint?**
Disallows null values.
7. **What is CHECK constraint?**
Ensures values meet a specific condition.
8. **What is DEFAULT constraint?**
Assigns a default value if none is given.
9. **What is composite primary key?**
A primary key with multiple columns.
10. **Can a table have multiple primary keys?**
No, but it can have a composite key.
11. **Can a table have multiple unique constraints?**
Yes.
12. **Difference between UNIQUE and PRIMARY KEY?**
Primary key cannot have nulls; unique can.
13. **Can a foreign key be NULL?**
Yes, unless NOT NULL is specified.
14. **What is ON DELETE CASCADE?**
Deletes related child rows when parent is deleted.
15. **What is ON UPDATE CASCADE?**
Updates child key when parent key changes.
16. **Can CHECK have multiple conditions?**
Yes, using AND/OR.
17. **Can CHECK use subqueries?**
Usually not supported.
18. **Can constraints be disabled?**
Yes, using ALTER TABLE.
19. **Can we name constraints?**
Yes, when creating or altering tables.
20. **Can we drop constraints?**
Yes, using ALTER TABLE DROP CONSTRAINT.

**Section 6: Indexing (Q101–Q120)**

1. **What is an index?**
Structure to speed up data retrieval.
2. **Types of indexes?**
Single-column, composite, unique, non-unique, full-text, bitmap.
3. **Does primary key create index automatically?**
Yes.
4. **Does unique constraint create index automatically?**
Yes.
5. **What is a clustered index?**
Stores data in the order of the index.
6. **What is a non-clustered index?**
Stores separate structure with pointers to data.
7. **Can we have multiple clustered indexes?**
No.
8. **Can we have multiple non-clustered indexes?**
Yes.
9. **What is a full-text index?**
Optimized for searching large text.
10. **What is a bitmap index?**
Uses bitmaps for low-cardinality data.
11. **What is a covering index?**
Contains all columns needed by a query.
12. **What is a filtered index?**
Built on a subset of rows.
13. **What is a function-based index?**
Built on the result of an expression.
14. **Advantages of indexes?**
Faster retrieval.
15. **Disadvantages of indexes?**
Slower writes and extra storage.
16. **How to create index?**
CREATE INDEX name ON table(col);
17. **How to drop index?**
DROP INDEX name;
18. **When to avoid indexes?**
On small or frequently updated tables.
19. **Does index use storage?**
Yes.
20. **How to view indexes on a table?**
SHOW INDEXES FROM table; (MySQL).

**Section 7: Transactions, ACID, Concurrency (Q121–Q160)**

1. **What is a transaction?**
A group of SQL operations executed as a single logical unit.
2. **What does ACID stand for?**
Atomicity, Consistency, Isolation, Durability.
3. **What is Atomicity?**
All operations complete successfully or none at all.
4. **What is Consistency?**
Database remains valid before and after the transaction.
5. **What is Isolation?**
Transactions don’t interfere with each other’s operations.
6. **What is Durability?**
Committed changes persist even after failures.
7. **What is COMMIT?**
Saves all changes made in a transaction.
8. **What is ROLLBACK?**
Reverts changes in a transaction.
9. **What is SAVEPOINT?**
Marks a point in a transaction to roll back to.
10. **What is autocommit?**
Each statement is automatically committed after execution.
11. **What is a dirty read?**
Reading uncommitted data from another transaction.
12. **What is a non-repeatable read?**
Same row read twice gives different results.
13. **What is a phantom read?**
A repeated query returns new rows.
14. **Read Uncommitted level?**
Allows dirty reads.
15. **Read Committed level?**
Only committed data is read.
16. **Repeatable Read level?**
Prevents non-repeatable reads.
17. **Serializable level?**
Highest isolation; prevents all anomalies.
18. **Trade-off of higher isolation?**
Better consistency, less concurrency.
19. **What is a lock?**
Mechanism to control concurrent access.
20. **What is a shared lock?**
Allows read, blocks write.
21. **What is an exclusive lock?**
Blocks both reads and writes from others.
22. **What are intention locks?**
Indicate future locking plans.
23. **Row-level vs table-level locks?**
Row-level is finer; table-level is coarser.
24. **What is a deadlock?**
Two transactions waiting on each other’s locks.
25. **How to resolve deadlocks?**
Detect and abort one transaction.
26. **What is blocking?**
Waiting due to a lock, but not a deadlock.
27. **Optimistic vs pessimistic concurrency?**
Optimistic checks for conflicts at commit; pessimistic locks early.
28. **What is MVCC?**
Multi-Version Concurrency Control – uses row versions.
29. **What is snapshot isolation?**
Reads from a consistent snapshot.
30. **What is write skew?**
Concurrent writes violate constraints under snapshot isolation.
31. **What is an idempotent operation?**
Can be repeated without changing the result.
32. **Why retryable transactions?**
To recover from deadlocks or serialization failures.
33. **Impact of long transactions?**
Hold locks and consume resources longer.
34. **When to disable autocommit?**
For multi-step transactions.
35. **How to handle partial failures?**
Use SAVEPOINTS and rollback only the failed part.
36. **Why use read-only transactions?**
For optimization and safety.
37. **Purpose of NOWAIT/SKIP LOCKED?**
Avoid waiting on locked rows.
38. **What is transactional DDL?**
DDL inside a transaction (DB-dependent).
39. **What is two-phase commit (2PC)?**
Ensures atomic commit across multiple databases.
40. **Risks of distributed transactions?**
Latency, complexity, and blocking.

**Section 8: Views & Materialized Views (Q161–Q180)**

1. **What is a view?**
A stored SQL query acting as a virtual table.
2. **Advantages of views?**
Simplifies queries, adds security, reusability.
3. **What is an updatable view?**
Allows inserts/updates to base tables.
4. **Can all views be updated?**
No; joins/aggregates often prevent updates.
5. **What is WITH CHECK OPTION?**
Prevents modifications outside view’s filter.
6. **What is a materialized view?**
Stores query results physically.
7. **Difference between view and materialized view?**
View is virtual; materialized view stores data.
8. **Types of materialized view refresh?**
Complete, fast, on-demand, scheduled.
9. **Why use materialized views?**
Improve performance for complex queries.
10. **Trade-offs of materialized views?**
Storage and refresh cost.
11. **How can views enforce security?**
Restrict access to certain columns/rows.
12. **What is schema binding?**
Prevents underlying schema changes.
13. **What is an indexed view?**
A view with indexes for performance.
14. **How to implement row-level security with views?**
Add WHERE clauses filtering by user.
15. **Possible view performance issue?**
Nested views can be slow.
16. **What is a temporary view?**
Session-specific logical view.
17. **Effect of renaming/dropping a view?**
Breaks dependent queries.
18. **Are parameterized views supported?**
Not in standard SQL; use table functions.
19. **How is stale data seen in materialized views?**
It’s as fresh as the last refresh.
20. **How to refresh MV without blocking reads?**
Use concurrent refresh methods.

**Section 9: Stored Procedures & User-Defined Functions (Q181–Q210)**

1. **What is a stored procedure?**
A set of SQL statements stored in the database and executed as a unit.
2. **What is a user-defined function (UDF)?**
A routine that accepts inputs and returns a value, can be used in queries.
3. **Procedure vs function difference?**
Functions must return a value; procedures may not.
4. **What is a deterministic function?**
Always returns the same output for the same input.
5. **Why are side-effect-free functions important?**
They allow better optimization and predictability.
6. **Scalar vs table-valued functions?**
Scalar returns one value; table-valued returns a table.
7. **What are parameter modes?**
IN, OUT, INOUT.
8. **How to handle exceptions in procedures/functions?**
Use TRY...CATCH or equivalent in the DBMS.
9. **Why use stored procedures?**
Reusability, security, and reduced network traffic.
10. **Downside of overusing stored procedures?**
Harder to maintain, test, and port between systems.
11. **Performance pitfalls of UDFs?**
Row-by-row execution can be slow; prefer set-based logic.
12. **What is bulk processing?**
Performing operations on multiple rows in one go.
13. **Dynamic SQL in a procedure?**
Allows flexible queries but must be secured against SQL injection.
14. **Return codes vs result sets?**
Return codes for status; result sets for data.
15. **What is privilege separation in procedures?**
Grant execute permission to users, not direct table access.
16. **Logging inside a procedure?**
Insert into an audit table for tracking.
17. **What is idempotent upsert?**
An insert/update that can run multiple times without side effects.
18. **Using sequences in procedures/functions?**
For generating unique keys.
19. **What are default parameters?**
Parameters with default values if not provided.
20. **Returning multiple result sets?**
Supported in some DBs (e.g., SQL Server, MySQL).
21. **How to version procedures?**
Use version suffixes and migration scripts.
22. **Testing stored procedures?**
Use test data and rollback after execution.
23. **Security definer vs invoker?**
Definer runs with creator’s privileges; invoker uses caller’s.
24. **Handling timeouts in long procedures?**
Use batching and adjust client/server timeout settings.
25. **When to use recursive functions?**
For hierarchical or tree structures.
26. **Why prefer set-based operations over row-by-row?**
Set operations are faster and more efficient.
27. **What is instrumentation in procedures?**
Collect timing, row counts, and error logs.
28. **How to prevent SQL injection in procedures?**
Use bind parameters and avoid string concatenation.
29. **What are cross-database calls?**
Executing procedures in another database.
30. **Best practice for deploying procedures/UDFs?**
Use version control and tested deployment scripts.

**Section 10: Triggers & Scheduled Events (Q211–Q230)**

1. **What is a trigger?**
Code that runs automatically in response to table events.
2. **BEFORE vs AFTER trigger?**
BEFORE runs before the change; AFTER runs after.
3. **Row-level vs statement-level triggers?**
Row-level runs once per row; statement-level runs once per statement.
4. **What is a mutating table error?**
Occurs when reading from the same table in its trigger.
5. **What is an audit trigger?**
Captures who changed what and when.
6. **What is a soft delete trigger?**
Marks a row as deleted instead of physically removing it.
7. **Cascading triggers risk?**
Can cause loops and performance issues.
8. **What is an INSTEAD OF trigger?**
Allows modifying views.
9. **Example of a preventive trigger?**
Blocking updates outside business hours.
10. **Trigger vs application logic?**
Keep simple integrity in triggers; complex logic in applications.
11. **What is a scheduler/event?**
Runs jobs on a set schedule within the DB.
12. **Examples of maintenance jobs?**
Refresh stats, refresh materialized views, archive old data.
13. **Can DB send email directly?**
Some can, but often handled by application services.
14. **Trigger performance impact?**
Triggers add overhead to DML operations.
15. **How to handle errors in triggers?**
Raise an error or log it.
16. **Security triggers?**
Enforce row-level security by filtering users.
17. **ID generation in triggers?**
Assigns keys before insert using sequences.
18. **Time issues in triggers?**
Use server time functions, not client time.
19. **When to disable triggers?**
During bulk loads or migrations.
20. **Testing triggers?**
Perform controlled DML and verify expected results.

**Section 11: Advanced Querying (Q231–Q270)**

1. **What is a subquery?**
A query inside another query.
2. **What is a correlated subquery?**
Subquery that references columns from the outer query.
3. **What is a CTE (WITH)?**
A temporary named result set for readability and reuse.
4. **Why use recursive CTEs?**
For hierarchical or recursive data.
5. **What is a window function?**
Performs calculations across a set of rows without collapsing them.
6. **Purpose of OVER(PARTITION BY ... ORDER BY ...)?**
Defines the window for the calculation.
7. **What is ROW\_NUMBER()?**
Assigns a unique sequential number to rows.
8. **RANK() vs DENSE\_RANK()?**
RANK leaves gaps; DENSE\_RANK doesn’t.
9. **What are LAG and LEAD?**
Access previous/next row values.
10. **What are FIRST\_VALUE and LAST\_VALUE?**
Return the first or last value in the window.
11. **What does NTILE(n) do?**
Divides rows into n groups.
12. **How to calculate moving average?**
AVG with a ROWS frame.
13. **How to calculate cumulative sum?**
SUM with UNBOUNDED PRECEDING.
14. **How to get top-N per group?**
Use ROW\_NUMBER and filter to 1.
15. **What is pivot?**
Converts rows into columns.
16. **What is unpivot?**
Converts columns into rows.
17. **EXISTS vs IN difference?**
EXISTS checks for existence; IN checks for matching values.
18. **ANY vs ALL difference?**
ANY matches one; ALL matches all.
19. **Where can scalar subqueries be used?**
SELECT, WHERE, HAVING clauses.
20. **What is a derived table?**
Subquery in the FROM clause.
21. **How to do an anti-join?**
LEFT JOIN with IS NULL or NOT EXISTS.
22. **How to do a semi-join?**
Use EXISTS.
23. **What are set operations?**
UNION, UNION ALL, INTERSECT, EXCEPT.
24. **UNION vs UNION ALL performance?**
UNION removes duplicates, slower.
25. **What is gaps-and-islands problem?**
Finding continuous sequences and gaps in data.
26. **Top record per group without join?**
Use window functions.
27. **What is conditional aggregation?**
Aggregate with CASE inside.
28. **How to get percentiles?**
Use PERCENTILE\_CONT or equivalent.
29. **How to do sampling?**
TABLESAMPLE or random filtering.
30. **How to fill time-series gaps?**
Join with a calendar table.
31. **What is a sargable predicate?**
Allows index usage.
32. **Example of non-sargable?**
Wrapping column in a function in WHERE.
33. **How to expand array/JSON?**
Use UNNEST or JSON functions.
34. **What is a lateral join?**
Join that allows referencing left table in subquery.
35. **What is a window frame clause?**
Defines the subset of rows for window functions.
36. **Why use HAVING?**
Filter after GROUP BY.
37. **What is ROLLUP/CUBE?**
Creates subtotals and grand totals.
38. **What are GROUPING SETS?**
Define multiple groupings in one query.
39. **What is distinct within group aggregate?**
COUNT(DISTINCT col) etc.
40. **What is QUALIFY clause?**
Filters on window function results (DB-specific).

**Section 12: Performance & Query Plans (Q271–Q300)**

1. **What is an execution plan?**
Steps DB will take to run a query.
2. **Purpose of EXPLAIN/ANALYZE?**
See plan and execution stats.
3. **Index scan vs table scan?**
Index scan reads selected rows; table scan reads all.
4. **Seek vs scan?**
Seek jumps to data; scan reads sequentially.
5. **Join methods?**
Nested loop, hash join, merge join.
6. **What is cardinality estimate?**
Estimated row count in plan.
7. **Why are statistics important?**
Improve cardinality accuracy.
8. **What is parameter sniffing?**
Plan optimized for one parameter may not fit others.
9. **What is selectivity?**
Fraction of rows that match filter.
10. **Benefit of covering index?**
Satisfies query without reading table.
11. **Order of columns in composite index?**
Most selective first.
12. **When to index expressions?**
When filtering by computed values.
13. **What is over-indexing?**
Too many indexes slow writes.
14. **How to optimize OR conditions?**
Rewrite or use separate queries with UNION.
15. **Index with LIKE '%term'?**
Needs special indexes like full-text.
16. **Pagination performance tip?**
Use seek method instead of OFFSET.
17. **What is N+1 query problem?**
Repeated queries for related data.
18. **When to materialize intermediate results?**
For complex queries with reuse.
19. **Temp table vs CTE?**
Temp table persists for session; CTE is inline.
20. **What are optimizer hints?**
Directives to influence plan choice.
21. **What is a parallel query?**
Uses multiple processors.
22. **I/O-bound vs CPU-bound?**
Based on resource bottleneck.
23. **What is a hotspot index?**
Many inserts into same page.
24. **What is fill factor?**
Space left in index pages.
25. **Why parameterize queries?**
Plan reuse and security.
26. **CTE for readability vs performance?**
Good for readability; perf varies.
27. **When to denormalize?**
For read-heavy workloads.
28. **Why cache queries?**
Reduce repeated execution.
29. **What is slow query log?**
Captures queries exceeding a time limit.
30. **What is workload profiling?**
Analyzing query patterns and resource use.

**Section 13: DBA Basics – Security, Backup/Restore (Q301–Q330)**

1. **User vs role?**
User = identity; role = permission set.
2. **What is least privilege principle?**
Grant only necessary permissions.
3. **What do GRANT and REVOKE do?**
Give or remove permissions.
4. **What is row-level security?**
Filters rows based on user.
5. **What is column masking?**
Hides part or all of a sensitive column.
6. **Why audit logging?**
For compliance and tracking changes.
7. **Encryption at rest vs in transit?**
At rest = stored data; in transit = data over network.
8. **Types of backups?**
Full, incremental, differential, logical, physical.
9. **What is point-in-time recovery?**
Restoring to a specific time.
10. **Why verify backups?**
To ensure they can be restored.
11. **RPO vs RTO?**
Recovery point vs recovery time.
12. **Hot vs warm vs cold standby?**
Hot = live; warm = delayed; cold = offline.
13. **Password policy?**
Rules for complexity and expiry.
14. **Service account best practice?**
Least privilege, no interactive logins.
15. **Schema vs database?**
Schema = namespace; database = logical storage.
16. **Why schedule maintenance windows?**
To run heavy tasks during low usage.
17. **What to monitor in DB?**
CPU, memory, I/O, locks, slow queries.
18. **How to set alert thresholds?**
Based on baselines and SLOs.
19. **Why use connection pooling?**
Reuse connections to save time.
20. **Risk of too many connections?**
Resource exhaustion.
21. **What is a resource governor?**
Controls resource use per workload.
22. **Safe schema changes?**
Backward-compatible migrations.
23. **What is online index rebuild?**
Rebuild without blocking queries.
24. **What is data retention policy?**
Rules for how long data is kept.
25. **What is anonymization/tokenization?**
Masking sensitive data.
26. **What is change data capture (CDC)?**
Capturing row changes for replication or ETL.
27. **Why use read replicas?**
Offload reads from primary.
28. **Failover vs switchover?**
Failover = unplanned; switchover = planned.
29. **What is split-brain?**
Two primaries active at once.
30. **What is a runbook?**
Step-by-step incident procedure.

**Section 14: Data Modeling, Normalization, Partitioning, JSON (Q331–Q350)**

1. **What is normalization?**
Organizing data to reduce redundancy.
2. **What is 1NF?**
Atomic values, no repeating groups.
3. **What is 2NF?**
1NF + remove partial dependencies.
4. **What is 3NF?**
2NF + remove transitive dependencies.
5. **What is BCNF?**
Stronger form of 3NF.
6. **When to denormalize?**
For performance in read-heavy cases.
7. **What is an ER diagram?**
Visual representation of entities and relationships.
8. **What is cardinality?**
Defines relationship type (1:1, 1:M, M:N).
9. **Surrogate vs natural key?**
Surrogate = artificial; natural = from business data.
10. **What is partitioning?**
Splitting a table into smaller parts.
11. **When to use range partitioning?**
For time-series data.
12. **When to use hash partitioning?**
For even data distribution.
13. **When to use list partitioning?**
For specific categories.
14. **Sharding vs partitioning?**
Sharding = across servers; partitioning = within one.
15. **Replication vs sharding?**
Replication copies data; sharding splits it.
16. **Does SQL support JSON?**
Yes, in many databases.
17. **Why index JSON/XML?**
To speed up searches inside documents.
18. **Best practice for time zones?**
Store in UTC, convert on display.
19. **Sequences vs identity columns?**
Sequence = independent; identity = tied to a table.
20. **Soft delete vs hard delete?**
Soft = mark as deleted; hard = remove row.