**Section 1: SQL Basics (Q1–Q20)**

1. **SQL भनेको के हो?**
Structured Query Language — Database मा डेटा Access, Manage र Modify गर्न प्रयोग हुने भाषा।
2. **SQL र MySQL बीच के फरक छ?**
SQL भाषा हो; MySQL एक Database Management System हो जसले SQL प्रयोग गर्छ।
3. **Database भनेको के हो?**
डेटा व्यवस्थित रूपमा संग्रह गर्ने स्थान।
4. **DBMS भनेको के हो?**
Database Management System — Database बनाउन, राख्न र Access गर्नको लागि Software।
5. **RDBMS भनेको के हो?**
Relational DBMS — Table मा Row र Column को रूपमा डेटा राख्ने DBMS।
6. **SQL का मुख्य प्रकारका Statements के हुन्?**
DDL, DML, DCL, TCL।
7. **DDL भनेको के हो?**
Data Definition Language — Table Structure बनाउने, Change गर्ने (जस्तै: CREATE, ALTER, DROP)।
8. **DML भनेको के हो?**
Data Manipulation Language — डेटा Insert, Update, Delete गर्ने (जस्तै: INSERT, UPDATE, DELETE)।
9. **DCL भनेको के हो?**
Data Control Language — Access Permission दिने वा हटाउने (जस्तै: GRANT, REVOKE)।
10. **TCL भनेको के हो?**
Transaction Control Language — Transaction Manage गर्ने (जस्तै: COMMIT, ROLLBACK, SAVEPOINT)।
11. **SQL मा Table भनेको के हो?**
Rows र Columns मा डेटा राख्ने Structure।
12. **Row र Column के हुन्?**
Row = Record; Column = Field।
13. **SQL मा Primary Key के हो?**
Unique र Not Null भएको Column।
14. **SQL मा Foreign Key के हो?**
अर्को Table को Primary Key लाई Reference गर्ने Column।
15. **SQL मा NULL value के हो?**
Value नभएको Field (Empty नभएर Null)।
16. **SELECT Statement को प्रयोग के हो?**
Table बाट डेटा Retrieve गर्न।
17. **SQL मा Aliases के हो?**
Column वा Table लाई Temporary Name दिनु।
18. **DISTINCT keyword को प्रयोग के हो?**
Duplicate हटाएर Unique Rows देखाउन।
19. **ORDER BY clause के हो?**
Result Ascending वा Descending मा Sort गर्न।
20. **WHERE clause के हो?**
Specific Condition अनुसार डेटा Filter गर्न।

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

1. **AND र OR बीच के फरक छ?**
AND — दुवै Condition True हुनुपर्छ; OR — कम्तीमा एक True भए पुग्छ।
2. **BETWEEN clause के हो?**
Range भित्रको डेटा Filter गर्न।
3. **IN clause के हो?**
Multiple Values मा Match गर्न।
4. **LIKE clause के हो?**
Pattern Matching गर्न।
5. **% र \_ wildcard के हुन्?**
% = Multiple Characters, \_ = Single Character Match।
6. **IS NULL र IS NOT NULL बीच फरक?**
Null Value छ कि छैन जाँच्ने।
7. **Comparison Operators SQL मा के हुन्?**
=, !=, <>, >, <, >=, <=।
8. **Logical Operators SQL मा के हुन्?**
AND, OR, NOT।
9. **Arithmetic Operators SQL मा के हुन्?**
+, -, \*, /।
10. **NOT keyword के हो?**
Condition को उल्टो Result दिन्छ।
11. **ALL keyword के हो?**
सब भन्दा Compare गर्ने।
12. **ANY keyword के हो?**
कम्तीमा एक Value सँग Compare गर्ने।
13. **EXISTS keyword के हो?**
Subquery मा डेटा छ कि छैन जाँच्ने।
14. **UNION र UNION ALL बीच फरक?**
UNION — Duplicate हटाउँछ; UNION ALL — Duplicate राख्छ।
15. **INTERSECT clause के हो?**
दुवै Query मा Common Rows निकाल्ने।
16. **MINUS clause के हो?**
पहिलो Query मा मात्र हुने Rows देखाउने।
17. **CASE statement के हो?**
IF-ELSE logic SQL मा प्रयोग गर्ने।
18. **COALESCE function के हो?**
पहिलो Non-Null Value फर्काउने।
19. **NULLIF function के हो?**
दुई Value बराबर भए Null फर्काउने।
20. **CAST र CONVERT बीच फरक?**
Data Type बदल्ने Functions।

**Section 3: Joins (Q41–Q60)**

1. **Join भनेको के हो?**
Multiple Tables बाट Related Data Combine गर्ने।
2. **INNER JOIN के हो?**
दुवै Table मा Match भएको Data मात्र फर्काउने।
3. **LEFT JOIN के हो?**
Left Table को सबै Data + Right Table मा Match भए मात्र।
4. **RIGHT JOIN के हो?**
Right Table को सबै Data + Left Table मा Match भए मात्र।
5. **FULL OUTER JOIN के हो?**
दुवै Table को सबै Data, Match नभएकोमा NULL।
6. **CROSS JOIN के हो?**
Cartesian Product, सबै Combination फर्काउने।
7. **SELF JOIN के हो?**
एउटै Table मा Join गर्ने।
8. **NATURAL JOIN के हो?**
Same Column Name भएको Data Match गर्ने।
9. **EQUI JOIN के हो?**
Equality Condition प्रयोग गर्ने Join।
10. **NON-EQUI JOIN के हो?**
Non-equality Condition प्रयोग गर्ने Join।
11. **Theta Join के हो?**
Any Condition Join (>, <, >=, <=)।
12. **Hash Join के हो?**
Hash Table प्रयोग गरेर Fast Join।
13. **Merge Join के हो?**
Sorted Tables Merge गरेर Join गर्ने।
14. **Anti Join के हो?**
Not Matching Data मात्र फर्काउने।
15. **Semi Join के हो?**
Match भएको Data मात्र फर्काउने, Duplicate हटाएर।
16. **Join vs Subquery?**
Join — Tables Combine; Subquery — Nested Query।
17. **Can we join more than two tables?**
हो, Multiple Joins प्रयोग गर्न सकिन्छ।
18. **Can we join without ON clause?**
हो, CROSS JOIN मा।
19. **Which join returns maximum rows?**
CROSS JOIN।
20. **Which join returns minimum rows?**
INNER JOIN।

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

1. **Aggregate Function भनेको के हो?**
Multiple Rows बाट Single Value फर्काउने Function (जस्तै: SUM, AVG)।
2. **Scalar Function भनेको के हो?**
प्रत्येक Row को लागि Single Value फर्काउने Function (जस्तै: UPPER, LOWER)।
3. **COUNT() function के हो?**
Rows को संख्या गन्ने।
4. **SUM() function के हो?**
Numeric Column को Total निकाल्ने।
5. **AVG() function के हो?**
Numeric Column को औसत निकाल्ने।
6. **MIN() function के हो?**
Column को सबैभन्दा सानो Value फर्काउने।
7. **MAX() function के हो?**
Column को सबैभन्दा ठूलो Value फर्काउने।
8. **ROUND() function के हो?**
Decimal Number लाई Round गर्ने।
9. **CEIL() र FLOOR() बीच फरक?**
CEIL — अगाडिको Integer; FLOOR — पछाडिको Integer।
10. **ABS() function के हो?**
Absolute Value फर्काउने।
11. **POWER() function के हो?**
एक Number को Power निकाल्ने।
12. **MOD() function के हो?**
Division पछि Remaining Value फर्काउने।
13. **UPPER() function के हो?**
String लाई Uppercase मा बदल्ने।
14. **LOWER() function के हो?**
String लाई Lowercase मा बदल्ने।
15. **INITCAP() function के हो?**
प्रत्येक शब्दको पहिलो अक्षर Capital गर्ने।
16. **LTRIM() function के हो?**
String को Leading Characters हटाउने।
17. **RTRIM() function के हो?**
String को Trailing Characters हटाउने।
18. **TRIM() function के हो?**
String को दुवै तिरको Unwanted Characters हटाउने।
19. **SUBSTR() function के हो?**
String को Substring निकाल्ने।
20. **REPLACE() function के हो?**
String को Part अर्को Value सँग Replace गर्ने।

**Section 5: Constraints (Q81–Q100)**

1. **Constraint भनेको के हो?**
Data Integrity कायम राख्ने Rule।
2. **Types of Constraints के के हुन्?**
PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK, DEFAULT।
3. **PRIMARY KEY constraint के हो?**
Unique र Not Null Value भएको Column।
4. **FOREIGN KEY constraint के हो?**
अर्को Table को PRIMARY KEY लाई Reference गर्ने Column।
5. **UNIQUE constraint के हो?**
Duplicate Value रोक्ने।
6. **NOT NULL constraint के हो?**
Null Value रोक्ने।
7. **CHECK constraint के हो?**
Specific Condition पूरा नभए Value Insert हुन नदिने।
8. **DEFAULT constraint के हो?**
Value नदिए Default Value राख्ने।
9. **Composite Primary Key के हो?**
Multiple Columns मिलेर बनेको Primary Key।
10. **Can a table have multiple PRIMARY KEYs?**
हुँदैन, तर Composite Primary Key बनाउन सकिन्छ।
11. **Can a table have multiple UNIQUE constraints?**
हुन्छ।
12. **Difference between UNIQUE and PRIMARY KEY?**
PRIMARY KEY मा Null हुँदैन; UNIQUE मा Null हुन सक्छ।
13. **Can FOREIGN KEY be NULL?**
हुन्छ, यदि NOT NULL constraint छैन भने।
14. **ON DELETE CASCADE के हो?**
Parent Row Delete हुँदा Related Child Rows पनि Delete हुने।
15. **ON UPDATE CASCADE के हो?**
Parent Key Update हुँदा Child Key पनि Update हुने।
16. **CHECK constraint मा Multiple Conditions राख्न मिल्छ?**
हुन्छ, AND/OR प्रयोग गरेर।
17. **Can a CHECK constraint use subquery?**
DBMS मा निर्भर हुन्छ; धेरैमा अनुमति हुँदैन।
18. **Can constraints be disabled?**
हुन्छ, ALTER TABLE प्रयोग गरेर।
19. **Can we name constraints?**
हुन्छ, CREATE TABLE वा ALTER TABLE मा Constraint Name दिन सकिन्छ।
20. **Can we drop constraints?**
हुन्छ, ALTER TABLE DROP CONSTRAINT प्रयोग गरेर।

**Section 6: Indexing (Q101–Q120)**

1. **Index भनेको के हो?**
Table मा Data खोजाइ छिटो बनाउन Memory Structure।
2. **Types of Index के के हुन्?**
Single-column, Composite, Unique, Non-Unique, Full-text, Bitmap।
3. **Primary Key ले Automatic Index बनाउँछ?**
हुन्छ।
4. **Unique constraint ले Automatic Index बनाउँछ?**
हुन्छ।
5. **Clustered Index के हो?**
Table को Data Physical Order मा राख्ने Index।
6. **Non-clustered Index के हो?**
Data को अलग Structure मा Pointer राख्ने Index।
7. **Can a table have multiple clustered indexes?**
हुँदैन, एक मात्र हुन्छ।
8. **Can a table have multiple non-clustered indexes?**
हुन्छ।
9. **Full-text Index के हो?**
Large Text Search को लागि Index।
10. **Bitmap Index के हो?**
Low Cardinality Data को लागि Bit Mapping प्रयोग गर्ने Index।
11. **Covering Index के हो?**
Query मा चाहिने सबै Columns समावेश भएको Index।
12. **Filtered Index के हो?**
Specific Condition अनुसार Index।
13. **Function-based Index के हो?**
Expression वा Function को Result मा Index।
14. **Advantages of Index?**
Search Speed बढ्छ।
15. **Disadvantages of Index?**
Insert/Update/Delete मा Performance घट्छ।
16. **How to create index?**
CREATE INDEX index\_name ON table\_name(column\_name);
17. **How to drop index?**
DROP INDEX index\_name;
18. **When should you avoid indexes?**
Small tables वा frequent updates भएका tables मा।
19. **Does indexing use extra storage?**
हुन्छ।
20. **How to view indexes on a table?**
SHOW INDEXES FROM table\_name; (MySQL मा)।

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

1. **Transaction भनेको के हो?**
एउटै logical unit रूपमा चलाइने SQL कार्यहरूको समूह (BEGIN → काम → COMMIT/ROLLBACK)।
2. **ACID को full form?**
Atomicity, Consistency, Isolation, Durability।
3. **Atomicity के हो?**
सबै काम पूरा वा कुनै पनि नहोस् — बीचैमा नछोडिने।
4. **Consistency के हो?**
Transaction अघि/पछि Database valid rules मिल्नु।
5. **Isolation के हो?**
एक Transaction ले अर्कोको बीचको काम नदेखोस।
6. **Durability के हो?**
COMMIT भएपछि Data स्थायी रहन्छ (crash पछि पनि)।
7. **COMMIT के हो?**
Transaction का सबै परिवर्तन स्थायी गर्ने आदेश।
8. **ROLLBACK के हो?**
Transaction का परिवर्तनहरू उल्ट्याउने।
9. **SAVEPOINT के हो?**
Transaction भित्र checkpoint राख्ने, आंशिक ROLLBACK गर्न मिल्ने।
10. **Autocommit के हो?**
प्रत्येक statement पछि स्वतः commit हुने मोड।
11. **Dirty Read के हो?**
Commit नभएको डेटा पढ्नु।
12. **Non-Repeatable Read के हो?**
एउटै row पुनः पढ्दा value फेरिएको पाउनु।
13. **Phantom Read के हो?**
एउटै query दोहोरिँदा नयाँ rows थपिएर आउनु।
14. **Read Uncommitted level के गर्छ?**
Dirty reads सम्म अनुमति दिन्छ (सबैभन्दा ढिलो safety)।
15. **Read Committed level के गर्छ?**
Commit भइसकेको डेटा मात्र पढ्न दिन्छ (dirty read हटाउँछ)।
16. **Repeatable Read level के गर्छ?**
एउटै row पुनः पढ्दा value स्थिर — non-repeatable read हटाउँछ।
17. **Serializable level के हो?**
सबैभन्दा कडा — transactions लाई serial जस्तो व्यवहार गराउँछ (phantom पनि हट्छ)।
18. **Isolation level बढाउँदा trade-off के?**
Consistency बढ्छ, throughput/parallelism घट्न सक्छ।
19. **Lock भनेको के हो?**
Concurrent access नियन्त्रित गर्न ताल्चा — shared/exclusive।
20. **Shared Lock (S) के हो?**
Read को लागि — धेरैले share गर्न सक्छन्, write रोक्छ।
21. **Exclusive Lock (X) के हो?**
Write को लागि — अरू read/write रोक्छ।
22. **Intention Locks केका लागि?**
Hierarchical locking coordination (table vs row)।
23. **Row-level vs Table-level lock?**
Row-level fine-grained; table-level coarse, तर सस्तो manage।
24. **Deadlock के हो?**
दुई transactions एकले अर्कोको lock कुरिरहने circular wait।
25. **Deadlock समाधान कसरी?**
DBMS detection र victim abort; app-side: consistent lock order, छोटो transactions।
26. **Blocking के हो?**
Lock को कारण अर्को wait हुनु — deadlock होइन, तर delay।
27. **Optimistic vs Pessimistic concurrency?**
Optimistic: conflict दुर्लभ मान्छ, commit मा check; Pessimistic: अगाडि नै lock।
28. **MVCC के हो?**
Multi-Version Concurrency Control — readers लाई snapshots देखाएर blocking घटाउने।
29. **Snapshot isolation के हो?**
Query/transaction ले एकै समयको snapshot हेर्छ।
30. **Write skew समस्या के हो?**
Snapshot isolation मा constraint-violating concurrent writes सम्भव।
31. **Idempotent operation के हो?**
धेरै पटक चल्दा पनि अन्तिम नतिजा उस्तै — retries सुरक्षित।
32. **Retryable transaction किन?**
Deadlock/serialization failure मा पुन: प्रयास गर्न।
33. **Long transaction को असर?**
धेरै lock/versions राख्नुपर्ने, vacuum/cleanup ढिलो, bloat।
34. **Autocommit कहिले अफ गर्ने?**
Multiple statements logically same unit (batch insert/update)।
35. **Partial failure handle कसरी?**
SAVEPOINT; failed भाग ROLLBACK TO SAVEPOINT; बाकी commit।
36. **Read-only transaction किन प्रयोग?**
Optimization; कुछ DBs ले हलुका snapshot/lock प्रयोग गर्छन्।
37. **Nowait/skip locked का प्रयोग?**
Wait नगरोस् वा locked rows स्किप गरोस् — queues/workers मा उपयोगी।
38. **Transactional DDL के हो?**
DDL पनि transaction मा सामेल (DB specific) — fail भए rollback।
39. **Two-phase commit (2PC) के हो?**
अनेक DBs बीच atomic commit (prepare → commit)।
40. **Distributed transaction को जोखिम?**
Latency, coordinator failure, blocking — अक्सर saga/compensation प्रयोग।

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

1. **View के हो?**
Stored SELECT — virtual table।
2. **View को फाइदा?**
Simplify query, security (column/row masking), reuse।
3. **Updatable view के हो?**
Base table मा insert/update गर्न दिने (शर्त पूरा भएमा)।
4. **View मा insert/update सधैं मिल्छ?**
हुँदैन; aggregates/joins/derived columns भए रोकिन सक्छ।
5. **WITH CHECK OPTION के हो?**
View को filter बाहिर पर्ने row insert/update हुन नदिनु।
6. **Materialized View (MV) के हो?**
Query result physically store हुने view (refreshable)।
7. **MV vs View फरक?**
MV cached data राख्छ; सामान्य view हरेक पटक compute हुन्छ।
8. **MV Refresh types?**
COMPLETE (full), FAST/INCREMENTAL (changes मात्र), ON DEMAND/ON SCHEDULE।
9. **MV को उपयोग?**
Heavy aggregates/pre-joins cache गरेर reporting गति।
10. **MV को trade-off?**
Storage र refresh cost।
11. **View security कसरी?**
Base table direct deny, view मा grant।
12. **Schema binding (DB specific) के हो?**
View को dependency schema सुरक्षित — underlying change रोक्ने।
13. **Indexed view (DB specific) के हो?**
View मा index; performance पणि कडा नियम।
14. **Row-level security view बाट?**
WHERE owner\_id = SESSION\_USER() जस्ता predicates।
15. **View performance issue?**
Nested views ले complex plan बनाउँछ; flatten गर्नुपर्छ।
16. **Temporary view के हो?**
Session-scoped logical view (DB specific: temp view/CTE)।
17. **View rename/drop effect?**
Dependent queries टूट्न सक्छन्; dependency tracking आवश्यक।
18. **Parameterized view हुन्छ?**
Standard SQL मा छैन; workaround: table functions/CTE।
19. **MV stale data कसरी देखिन्छ?**
Last refresh सम्मको snapshot; freshness SLA define गर्नु।
20. **MV refresh concurrent read कसरी?**
Atomic swap/CONCURRENT REFRESH (DB specific)।

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

1. **Stored Procedure के हो?**
DB मा stored executable code (logic + SQL)।
2. **UDF (Function) के हो?**
Input लिएर value फर्काउने routine — SELECT मा प्रयोग हुन सक्छ।
3. **Procedure vs Function फरक?**
Function return value अनिवार्य; procedure ले optional OUT parameters वा no return।
4. **Deterministic function के हो?**
उस्तै input → उस्तै output; caching/indexing अनुकूल।
5. **Side-effect free function किन महत्वपूर्ण?**
Query optimization र predictability।
6. **Scalar vs Table-valued function?**
Scalar single value; table-valued resultset फर्काउँछ (DB specific)।
7. **Parameter modes?**
IN, OUT, INOUT (DB specific variation)।
8. **Exception handling procedure/function मा?**
TRY…CATCH/EXCEPTION blocks (DB dialect अनुसार)।
9. **Procedure किन प्रयोग?**
Reuse, security (business rules DB मा), network round-trips घटाउनु।
10. **Overuse को downside?**
Logic DB-bound, versioning/testing कठिन, portability घट्छ।
11. **UDF performance pitfalls?**
Row-by-row execution (RBAR) ले slow; set-based rewrite राम्रो।
12. **Bulk processing कसरी?**
Set-based SQL, bulk insert, array/batch parameters।
13. **Dynamic SQL procedure मा?**
Flexible queries; तर SQL injection को जोखिम — bind parameters प्रयोग।
14. **Return codes vs result sets?**
Small status -> return code; data -> result set।
15. **Privilege separation कसरी?**
EXECUTE permission procedure मा, tables direct deny।
16. **Logging/Auditing procedure भित्र?**
Separate audit table मा insert; transaction impact ध्यान।
17. **Idempotent upsert कसरी?**
MERGE / INSERT…ON CONFLICT / INSERT…ON DUPLICATE KEY (dialect अनुसार)।
18. **Sequence उपयोग function/proc मा?**
NEXTVAL/IDENTITY fetch गरेर keys generate।
19. **Default parameters?**
Supported in केही DBs; नभए NULL check गरेर default लागू।
20. **Return multiple result sets?**
केही DBs (SQL Server, MySQL) support; client driver capable हुनुपर्छ।
21. **Procedure versioning कसरी?**
Suffix with \_v2, \_v3; migration scripts maintain।
22. **Testing stored code कसरी?**
Unit tests with fixtures; transaction rollbacks post-test।
23. **Security definer vs invoker?**
Definer अधिकारले execute vs caller अधिकारले (DB specific)।
24. **Timeout/long-running proc handle?**
Batching, commit chunks, progress table, client timeout config।
25. **Recursive function कहिले?**
Hierarchies (org charts), but watch stack/perf।
26. **Set-based alternative किन प्राथमिक?**
DB engines set operationsमा optimized छन् — कम context switch।
27. **Instrumentation कसरी?**
Timing, row counts, error codes, plan capture।
28. **SQL injection रोक्ने नियम?**
Bind parameters, whitelist identifiers, avoid string concatenation।
29. **Cross-db calls?**
DB links/federation — latency/consistency सावधानी।
30. **Proc/UDF deployment best practice?**
Migration tool, code review, rollback scripts, semantic versioning।

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

1. **Trigger के हो?**
Table event (INSERT/UPDATE/DELETE) मा automatic चल्ने code।
2. **BEFORE vs AFTER trigger?**
BEFORE — value संशोधन/validation; AFTER — audit/logging आदि।
3. **Row-level vs Statement-level triggers?**
Row-level प्रत्येक row मा; statement-level एकपटक।
4. **Mutating table त्रुटि के हो?**
Trigger भित्र त्यही table पढ्दा inconsistency — workaround: temp store/statement trigger।
5. **Audit trigger केका लागि?**
Who/when/what changed tracking।
6. **Soft delete pattern trigger बाट?**
DELETE मा instead flag set (deleted\_at), वास्तविक delete नगर्ने।
7. **Cascade trigger सावधानी?**
Recursive cascade ले performance/loops — FK CASCADE राम्रो।
8. **INSTEAD OF trigger के हो?**
View मा write सक्षम गर्न (DB specific)।
9. **Preventive trigger उदाहरण?**
Business hours बाहिर updates रोक्ने।
10. **Trigger vs application logic?**
Simple invariants DB मा; complex workflows app मा राख्नु उत्तम।
11. **Scheduler/Event के हो?**
DB भित्र cron-like jobs चलाउने (DB specific)।
12. **Maintenance jobs के-के?**
Stats refresh, MV refresh, purge/archive, reindex।
13. **Email/notification DB बाट?**
Normally app/service ले; DB बाट direct कम प्रयोग।
14. **Trigger performance impact?**
प्रत्येक DML मा extra work — heavy logic नहाल्नु।
15. **Error handling trigger मा?**
Raise error to abort DML; else log table मा लेख्नु।
16. **Security trigger?**
Row-level security enforce (deny if owner mismatch)।
17. **Id generation trigger?**
Sequence/UUID assign BEFORE INSERT।
18. **Clock/timestamp issues?**
Use CURRENT\_TIMESTAMP/server time; client time मा भर नपर्नु।
19. **Disable trigger कहिले?**
Bulk load migrations — पछि enable र validate।
20. **Testing triggers?**
Specific DML + expected side-effects verify, rollback after test।

**Section 11: Advanced Querying (CTE, Window, Subqueries, Pivot) (Q231–Q270)**

1. **Subquery के हो?**
Query भित्रको Query — scalar/rowset।
2. **Correlated subquery के हो?**
Outer row अनुसार पुन: execute हुने subquery।
3. **CTE (WITH) के हो?**
Query को readable, reusable named subquery।
4. **Recursive CTE किन?**
Hierarchies/graphs traverse (parent-child)।
5. **Window function के हो?**
Aggregate over window/partition बिना rows collapse।
6. **OVER(PARTITION BY … ORDER BY …) के गर्छ?**
Group/window परिभाषित गरी running/ordered calc।
7. **ROW\_NUMBER() के हो?**
Ordered unique sequence per partition।
8. **RANK() vs DENSE\_RANK()?**
RANK gaps राख्छ; DENSE\_RANK continuous।
9. **LAG/LEAD के हो?**
पिछल्लो/अघिल्लो row को value access।
10. **FIRST\_VALUE/LAST\_VALUE के हो?**
Window भित्रको पहिलो/अन्तिम value।
11. **NTILE(n) के गर्छ?**
Rows लाई n buckets मा बाँड्छ।
12. **Moving average कसरी?**
AVG(value) OVER (ORDER BY t ROWS n PRECEDING)।
13. **Cumulative sum कसरी?**
SUM(value) OVER (ORDER BY t ROWS UNBOUNDED PRECEDING)।
14. **Top-N per group कसरी?**
ROW\_NUMBER() over (PARTITION BY grp ORDER BY metric) = 1 filter।
15. **Pivot के हो?**
Rows → columns मा values spread (DB specific PIVOT वा conditional agg)।
16. **Unpivot के हो?**
Columns → rows मा melt/normalize।
17. **EXISTS vs IN फरक?**
EXISTS presence check (semi join); IN set membership — NULL semantics भिन्न।
18. **ANY vs ALL?**
ANY: कम्तीमा एक सँग true; ALL: सबै सँग true।
19. **Scalar subquery कहाँ प्रयोग?**
SELECT list, WHERE, HAVING — single value constraints।
20. **Derived table के हो?**
FROM भित्रको subquery as table alias।
21. **Anti-join SQL कसरी?**
WHERE NOT EXISTS (…) वा LEFT JOIN … WHERE right IS NULL।
22. **Semi-join SQL कसरी?**
WHERE EXISTS (…) — duplicates हटाएर presence मात्र।
23. **Set operations कुन-कुन?**
UNION, UNION ALL, INTERSECT, EXCEPT/MINUS।
24. **UNION vs UNION ALL expense?**
UNION deduplicate (sort/hash) — costlier; UNION ALL छिटो।
25. **Gaps-and-islands pattern?**
LAG देखि group बदलिने स्थान पत्ता लगाएर contiguous ranges निकाल्ने।
26. **Greatest-N-per-group joinless?**
Window ROW\_NUMBER filter trick (Q244)।
27. **Conditional aggregation के हो?**
SUM(CASE WHEN cond THEN 1 ELSE 0 END) — pivot without pivot।
28. **Percentile कसरी?**
PERCENTILE\_CONT / APPROX\_PERCENTILE (DB specific)।
29. **Sampling कसरी?**
TABLESAMPLE/BERNOULLI/SYSTEM वा RAND() filter (DB specific)।
30. **Time-series gap fill कसरी?**
Calendar table/recursive CTE + LEFT JOIN।
31. **Sargable predicate के हो?**
Index-friendly predicate (column on left, no function wrapping)।
32. **Non-sargable example?**
WHERE DATE(col)=…; solution: col >= start AND col < end।
33. **Explode/arrays कसरी?**
JSON/ARRAY functions वा lateral join/UNNEST (DB specific)।
34. **Lateral join (CROSS APPLY) के हो?**
Left row context प्रयोग गर्ने rowset function join।
35. **Window frame clause के हो?**
ROWS/RANGE BETWEEN … — frame सीमाना।
36. **HAVING किन चाहिन्छ?**
GROUP BY पछि aggregate filter गर्न।
37. **ROLLUP/CUBE के हो?**
Multiple levels aggregates (subtotal/grand total)।
38. **GROUPING SETS के हो?**
Arbitrary aggregate groups set परिभाषित।
39. **Distinct within group aggregate?**
COUNT(DISTINCT col) वा SUM(DISTINCT col) (dialect अनुसार)।
40. **Qualify clause (DB specific) के हो?**
Window result filter (SELECT पछि) — Snowflake/BigQuery शैली।

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

1. **Execution plan के हो?**
DB ले query चलाउन बनाएको step-by-step strategy।
2. **Explain/EXPLAIN ANALYZE किन?**
Plan र runtime stats हेरेर tuning।
3. **Index scan vs table scan?**
Index scan selective predicatesमा; table scan सबै row पढ्छ।
4. **Seek vs Scan फरक?**
Seek targeted ranges; scan लगातार ब्लकहरू।
5. **Join methods के-के?**
Nested loop, Hash join, Merge join।
6. **Cardinality estimate के हो?**
Planner को अनुमानित rows — गलत भए plan खराब।
7. **Statistics किन महत्वपूर्ण?**
सही cardinality को लागि — नियमित analyze/auto-stats।
8. **Parameter sniffing समस्या?**
Cached plan एउटा parameterका लागि राम्रो, अरूमा खराब — उपाय: recompile, hints।
9. **Selectivity के हो?**
Predicate ले कति rows घटायो — उच्च selectivity राम्रो indexing।
10. **Covering index ले कसरी मद्दत गर्छ?**
Table lookup बिना index बाटै answer — I/O घट्छ।
11. **Composite index मा column order?**
सबसे selective अगाडि; predicates/ORDER BY heed।
12. **Index on expression कहिले?**
Filtering expression repeatedly — function-based index।
13. **Over-indexing समस्या?**
DML slow, storage बढी, planner confusion।
14. **OR predicates optimization?**
UNION ALL split, bitmap index, or rewrite to IN।
15. **LIKE '%term' सँग index?**
Leading wildcard non-sargable — full-text/GIN/trigram प्रयोग।
16. **Pagination performance?**
OFFSET large slow; seek-method (WHERE id > last\_id LIMIT n)।
17. **N+1 query समस्या?**
बारम्बार child fetch — solve: joins/IN bulk।
18. **Materialize intermediate results कहिले?**
Complex CTEs caching (DB specific), temp tables — plan simplify।
19. **Temporary table vs CTE?**
Temp table reusable/multiple scans; CTE inline (some DBs materialize)।
20. **Hints/optimizer directives?**
Join order/index/parallelism guide — last resort।
21. **Parallel query के हो?**
Multiple workers — CPU heavy scans/aggregatesमा फाइदा।
22. **I/O vs CPU bound query छुट्याउने?**
Wait events/metrics — buffer cache hit vs CPU time।
23. **Hotspot index के हो?**
Monotonic inserts same page — randomize keys/partition।
24. **Fill factor/BTREE bloat?**
Update/splitले बिस्तारै bloat — reindex/vacuum (DB specific)।
25. **Parameterize queries किन?**
Plan reuse, injection prevention, cache benefit।
26. **CTE for readability vs performance?**
Readability बढ्छ; performance dialect-dependent।
27. **Denormalization कहिले?**
Read-heavy analytics; सावधानी: consistency/ETL cost।
28. **Cache किन?**
Repeated queries; app/DB result cache — invalidation policy।
29. **Slow query log के हो?**
Threshold भन्दा ढिलो queries capture — tuning entry point।
30. **Workload profiling कसरी?**
Top queries by time/calls, plan stability, regression watch।

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

1. **User र Role फरक?**
User = login identity; Role = privilege set।
2. **Least privilege सिद्धान्त?**
आवश्यक मात्र अधिकार दिनु — सुरक्षा बढ्छ।
3. **GRANT र REVOKE के गर्छन्?**
Access दिनु/फिर्ता लिनु।
4. **Row-level security के हो?**
Policy ले प्रति-row access नियन्त्रण।
5. **Column masking के हो?**
Sensitive fields partially/fully hide।
6. **Audit logging किन?**
Compliance/forensics — who did what/when।
7. **Encryption at rest vs in transit?**
Disk/storage encryption vs TLS connections।
8. **Backup types के-के?**
Full, Incremental/Differential, Logical (dump), Physical।
9. **Point-in-time recovery कसरी?**
Base backup + WAL/binlog/redo logs replay to target timestamp।
10. **Backup verification किन?**
Restore test बिना backup असल भए/नभएको थाहै हुँदैन।
11. **RPO vs RTO?**
Recovery Point Objective (data loss window) vs Recovery Time Objective (downtime)।
12. **Hot/Warm/Cold standby के हो?**
Hot: near real-time; Warm: periodic; Cold: offline/offsite।
13. **User password policy?**
Complexity, expiry, lockout — DB/IdP स्तरमा।
14. **Service accounts best practice?**
Rotated secrets, least privilege, non-interactive।
15. **Schema vs Database फरक?**
Schema = object namespace; Database = logical DB (engine specific)।
16. **Maintenance windows किन?**
Reindex, vacuum, stats refresh, upgrades — low traffic बेला।
17. **Monitoring के-के?**
CPU, memory, I/O, connections, locks, replication lag, slow queries।
18. **Alerting thresholds कसरी?**
Baseline + SLO; actionable alerts मात्र।
19. **Connection pooling किन?**
Handshake cost घटाउन; app side pools (HikariCP आदि)।
20. **Max connections risk?**
Too many = thrash; too few = queue — right-size pool।
21. **Resource governor/workload mgmt?**
Noisy neighbor control — per-role/queue limits।
22. **Schema change safety?**
Backward-compatible migrations, blue-green deploy, online DDL।
23. **Online index rebuild/DDL?**
Lockless/low-lock operations (engine specific)।
24. **Data retention policy?**
TTL/archive per compliance (GDPR/PII)।
25. **Anonymization/Tokenization?**
PII सुरक्षित साझा/testing को लागि।
26. **Change data capture (CDC) के हो?**
Row changes stream/log — ETL/replication।
27. **Read replicas किन?**
Read scaling, reporting — eventual consistency ध्यान।
28. **Failover vs Switchover?**
Failover = unplanned; Switchover = planned role change।
29. **Split-brain समस्या?**
Dual primaries — quorum/fencing आवश्यक।
30. **Runbook किन बनाउने?**
Incident response steps — MTTR घट्छ।

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

1. **Normalization के हो?**
Redundancy घटाउने design नियमहरू।
2. **1NF के हो?**
Atomic values, no repeating groups।
3. **2NF के हो?**
1NF + partial dependency हटाइने (composite key मा)।
4. **3NF के हो?**
2NF + transitive dependency हटाइने।
5. **BCNF के हो?**
Stronger 3NF — every determinant is a key।
6. **Denormalization कहिले?**
Read performance/analytics; ETL को लागत बुझेर।
7. **ER diagram के हो?**
Entity, Attribute, Relationship को दृश्य नक्सा।
8. **Cardinality (1:1, 1:M, M:N) के हो?**
सम्बन्धको बहुलता — junction table ले M:N resolve।
9. **Surrogate vs natural key?**
Surrogate synthetic (sequence/UUID); natural business attribute।
10. **Partitioning के हो?**
Large table लाई भागमा split (range/list/hash) — manageability/Performance।
11. **Range partition suitable कहिले?**
Date/time series — pruning benefits।
12. **Hash partition कहिले?**
Uniform distribution for high-write workloads।
13. **List partition कहिले?**
Discrete categories (region codes आदि)।
14. **Sharding vs Partitioning?**
Sharding = across servers; partitioning = within a server/cluster।
15. **Replication vs Sharding?**
Replication copies data; sharding splits data।
16. **JSON support SQL मा?**
JSON columns, operators, indexes (GIN/trigram) — engine specific।
17. **XML/JSON indexing किन?**
Path/value queries छिटो बनाउन।
18. **Time zone handling best practice?**
Store UTC, convert at presentation; TIMESTAMP WITH TIME ZONE where available।
19. **Sequences vs Identity/Auto-increment?**
Sequences standalone objects; Identity tied to table column (engine व्यवहार फरक)।
20. **Soft delete vs Hard delete?**
Soft: flag/timestamp; Hard: row हटाइने — auditing/restore विचार गरेर चयन।