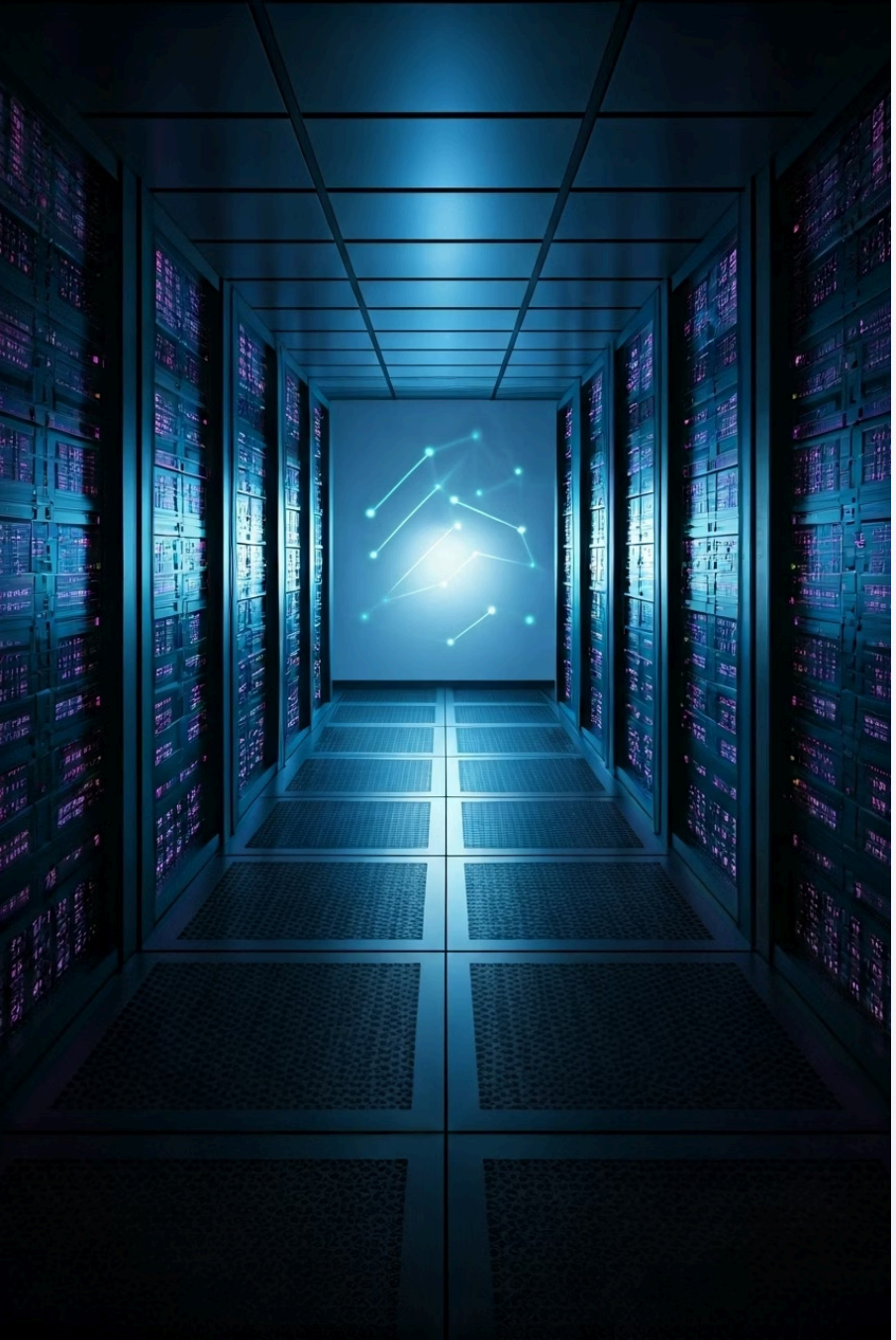
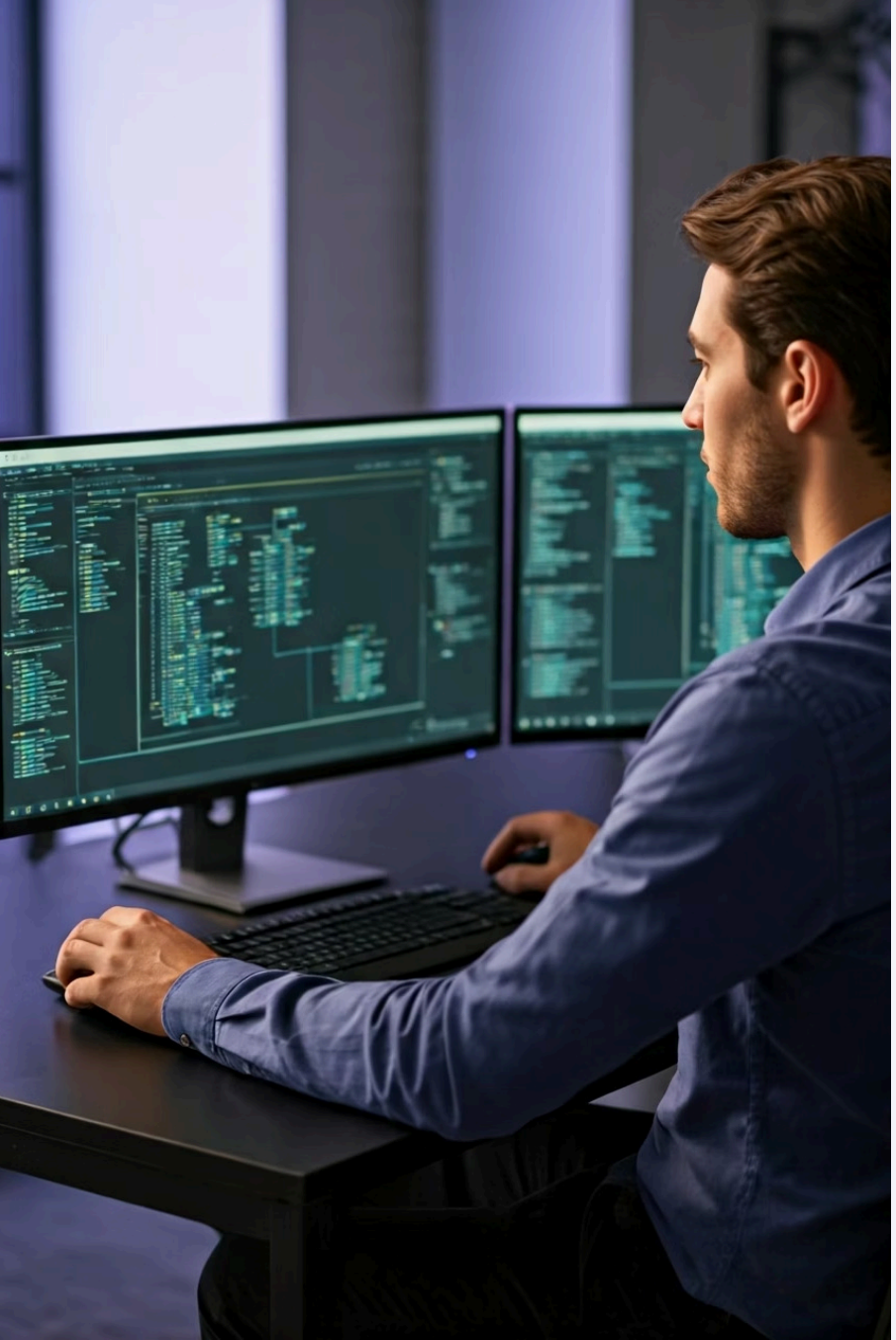


Advanced SQL: Multi-Table Operations, Transactions & Beyond

Master the art of joins, query optimization, ACID compliance, and programmatic SQL to build scalable, reliable database systems.



Introduction to Advanced SQL

Processing Multiple Tables

Joins, set operations, subqueries

Query Development

Efficiency, debugging, best practices

Transaction Integrity

ACID properties, concurrency control

Programmatic SQL

Triggers, stored routines, embedded SQL

Processing Multiple Tables

Join Types

INNER JOIN, LEFT/RIGHT JOIN, FULL OUTER JOIN, CROSS JOIN

Example: **SELECT Orders.OrderID, Customers.CustomerName FROM Orders INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;**

Set Operations

UNION, INTERSECT, EXCEPT (combine results vertically)

Subqueries

⚠️ Avoid Cartesian products! Always verify join conditions.

Tips for Developing Queries

Optimization Strategies

- Use EXPLAIN to analyze query plans
- Replace correlated subqueries with joins
- Minimize DISTINCT/GROUP BY where possible

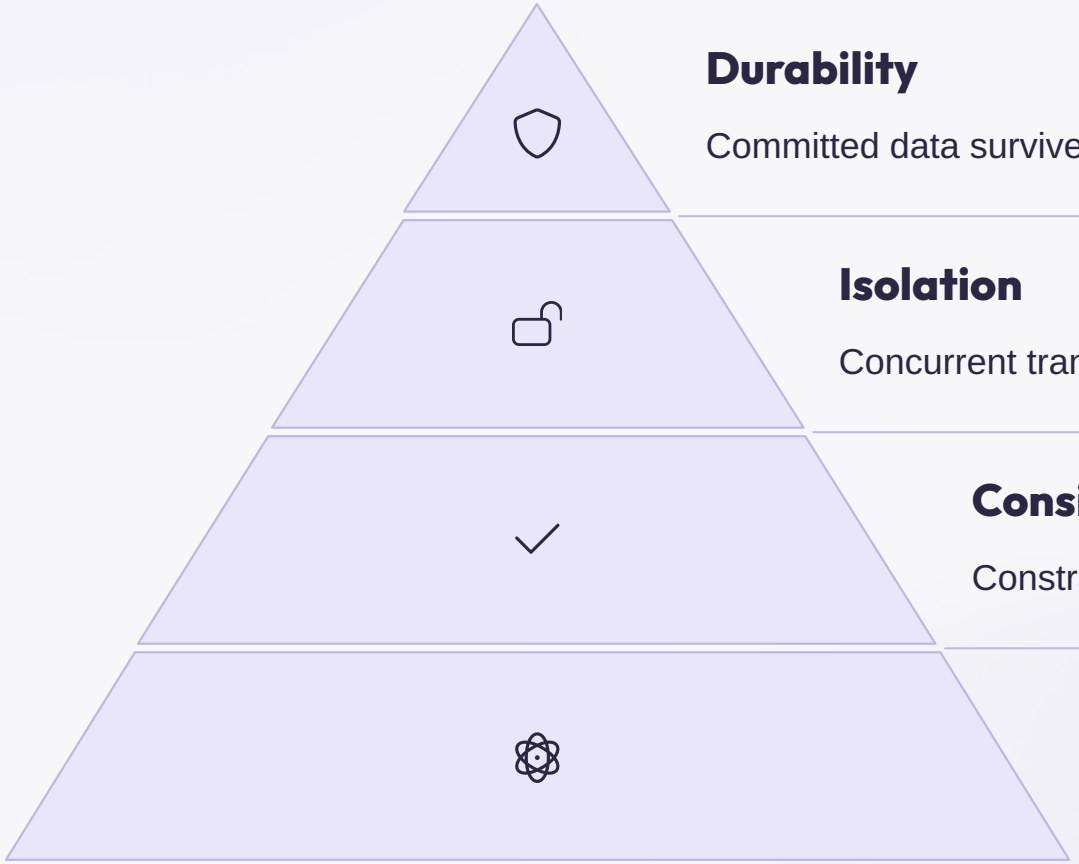
Debugging Techniques

- Build incrementally (test each JOIN/WHERE clause)
- Validate with sample data subsets
- Use CTEs for modularity

Pro Tip

"Write readable SQL – future you (and your team) will thank you!"

Ensuring Transaction Integrity



Durability

Committed data survives failures

Isolation

Concurrent transactions don't interfere

Consistency

Constraints preserve data validity

Atomicity

All-or-nothing execution

Example: START TRANSACTION; UPDATE Accounts SET Balance = Balance - 100 WHERE AccountID = 1; UPDATE Accounts SET Balance = Balance + 100 WHERE AccountID = 2; COMMIT;

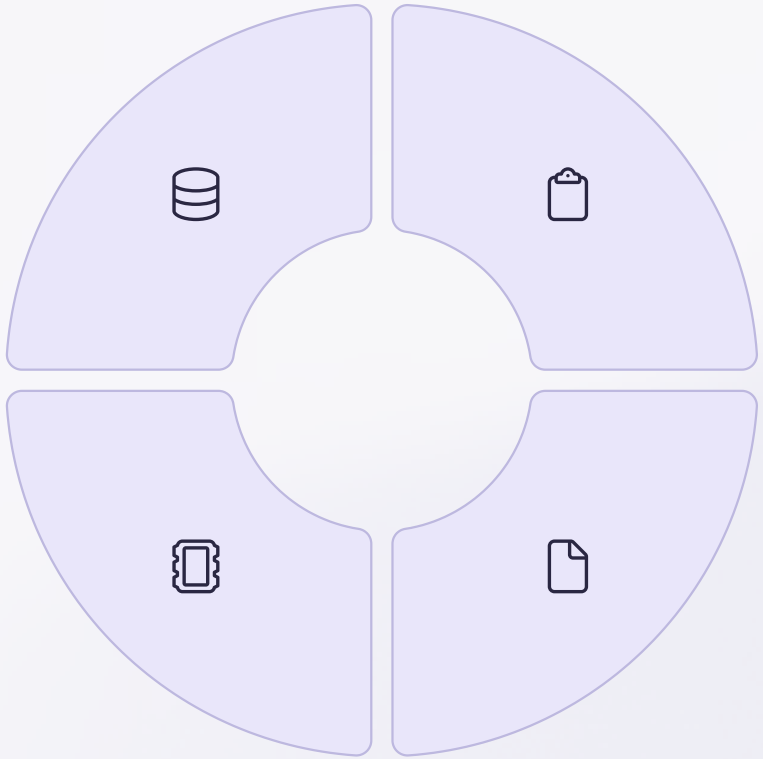
Data Dictionary Facility

Schema Discovery

Discover database structure dynamically

Key Tables

TABLES, COLUMNS, KEY_COLUMN_USAGE, ROUTINES



Governance

Enforce data policies

Documentation

Generate system documentation

SQL Enhancements & Extensions

JSON/XML Support

SELECT Product->>'$.name' FROM Orders; -- JSON extraction

Window Functions

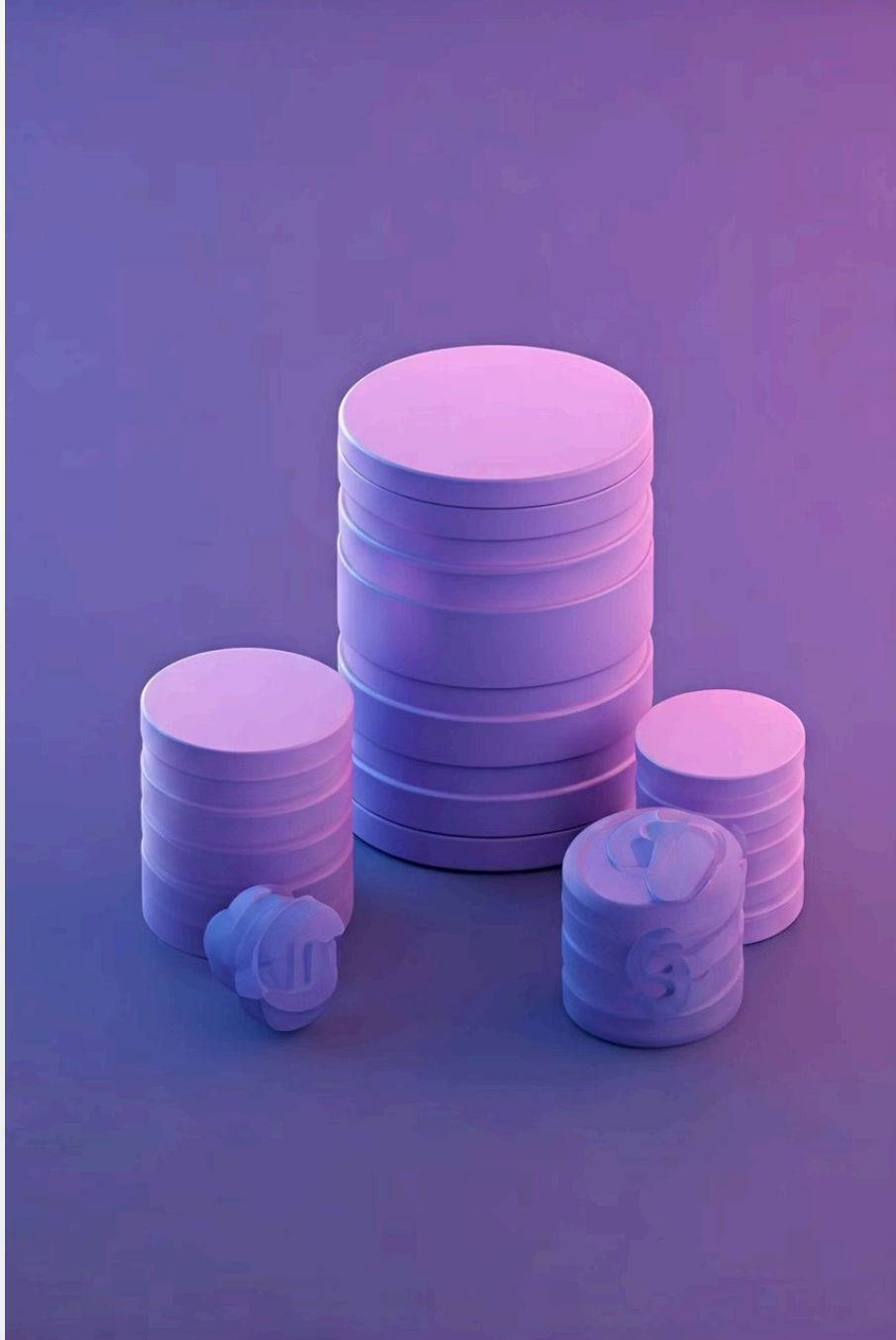
RANK() OVER (PARTITION BY CustomerID ORDER BY OrderDate)

GIS Extensions

Spatial data in PostGIS, MySQL

Vendor-Specific

PostgreSQL arrays, SQL Server T-SQL extensions



Triggers & Routines

Triggers

Automate actions on INSERT/UPDATE/DELETE

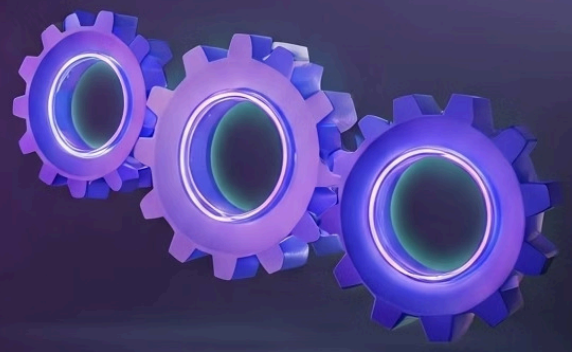
Procedures

Reusable logic blocks (CALL UpdateInventory(...))

Functions

Return values (e.g., CalculateTax(price))

Caution: Overuse can cause hidden performance bottlenecks!



Embedded SQL & Dynamic SQL

Embedded SQL

SQL statements hardcoded in host languages

Dynamic SQL

Construct queries at runtime

Security Note

Always sanitize inputs to prevent SQL injection!

Conclusion & Next Steps

Key Takeaways

- Joins > Subqueries for performance
- Transactions = Data integrity
- Metadata is your governance ally

Next Steps

- Explore ORMs vs. raw SQL
- Study cloud SQL variants
- Practice with engine-specific playgrounds

Questions?

We're here to help you master advanced SQL concepts!