Job Description: Principal Engineer (PostgreSQL Stored Procedures + AWS Aurora | Java on AWS Lambda)

Location: Kolkata | Seniority: Principal | Role Type: Full-time

Role Summary

We are hiring a Principal Engineer to lead a database-centric transactional platform built on AWS-managed services. The core of the system is Amazon Aurora (PostgreSQL-compatible), with significant business and analytical logic implemented as PostgreSQL stored procedures/functions. A thin compute layer—primarily Java running on AWS Lambda—exposes database routines through secure, well-designed APIs and orchestrates workflows.

This is a Principal-level role with strong emphasis on advanced SQL, stored procedure design, performance tuning, and production reliability in a fully managed AWS setup.

Key Responsibilities

Technical Leadership & Architecture

- Own the architecture for a managed AWS, database-first system (Aurora PostgreSQL plus Lambda-based Java services).
- Define and enforce engineering standards for PostgreSQL stored procedures/functions, schema design, and code quality.
- Establish release governance for database logic: versioning, deployment sequencing, rollback strategy, and change safety.
- Lead design reviews, mentor engineers, and act as an escalation point for complex technical and production issues.

PostgreSQL Stored Procedures & Advanced SQL Engineering

- Design, implement, and optimize PostgreSQL stored procedures/functions to encapsulate transactional and analytical logic.
- Define reusable patterns: parameterization, input validation, consistent error handling, auditing, and deterministic outputs.
- Own performance engineering: indexing strategy, query plan analysis, statistics maintenance, and concurrency/locking behavior.

- Create a disciplined approach to testing and safe rollout of stored procedure changes (golden datasets, regression suites, restore drills).

Java on AWS Lambda (API and Orchestration Layer)

- Build and review Java code running on AWS Lambda that exposes stored procedures via APIs (request validation, execution, response mapping).
- Define robust runtime patterns: retries, idempotency, timeouts, error mapping, and backpressure where applicable.
- Establish JDBC usage patterns suitable for serverless (connection strategy, pooling/proxy patterns as appropriate, transaction demarcation).
- Ensure least-privilege access to Aurora and secure handling of configuration and secrets.

Reliability, Operations & Observability (Managed AWS)

- Partner with platform/DevOps teams to ensure production readiness: monitoring, alerting, dashboards, and incident playbooks.
- Drive backup/restore and recovery drills; define RTO/RPO expectations and verify them through periodic exercises.
- Lead RCA for incidents, regressions, and performance degradations; implement durable fixes and preventative controls.
- Contribute to cost-aware design decisions while maintaining reliability and performance.

Must Have

- Principal-level experience building and operating database-centric backend systems with end-to-end ownership.
- Expert-level SQL skills on PostgreSQL (complex joins, window functions, CTEs, optimization, and concurrency considerations).
- Deep hands-on experience with PostgreSQL stored procedures/functions for core business logic and transactions.
- Strong understanding of RDBMS internals: ACID, isolation levels, locking/deadlocks, indexing, execution plans, and connection management.
- Strong Java backend fundamentals and ability to design a thin, reliable orchestration/API layer around database routines.
- Strong AWS fundamentals and production operations mindset (reliability, monitoring, incident response).
- Demonstrated technical leadership: architecture ownership, mentoring, and high-quality reviews with pragmatic trade-offs.

Should Have

- Hands-on experience with Amazon Aurora PostgreSQL (or RDS PostgreSQL), including operational characteristics and performance tuning.

- Experience running Java services on AWS Lambda (deployment/versioning, timeouts, cold starts, logging/tracing).
- Experience with database change management tooling (Flyway, Liquibase, or equivalent) including stored procedure versioning.
- Experience designing APIs around database routines: contract design, pagination, rate limiting, consistent error modeling, and SLAs.
- Observability discipline: structured logs, correlation IDs, metrics, dashboards, and alerting (CloudWatch/Datadog or similar).

Nice to Have

- Experience with event-driven/serverless orchestration (SQS, SNS, EventBridge, Step Functions) where it complements DB workflows.
- Experience standardizing SQL/stored-procedure engineering across teams (templates, linters, performance checklists, coding conventions).
- Experience building automated test harnesses for database routines (fixtures, golden datasets, regression automation).
- Exposure to security/compliance requirements (audit logs, access segregation, encryption, secrets management).

Experience & Qualification

- Typically 12–18+ years overall engineering experience, with a clear record of Principal-level impact.
- Bachelor's/Master's in Engineering/Computer Science or equivalent practical experience.

What Success Looks Like (First 90–120 Days)

- Define and roll out standards for PostgreSQL stored procedure development (patterns, testing, versioning, rollout/rollback).
- Improve performance predictability through query plan governance, indexing strategy, and concurrency/locking discipline.
- Establish a clean Java-on-Lambda API layer with consistent error handling, idempotency, and operational visibility.
- Reduce operational risk via documented recovery procedures, monitoring/alerting, and regular recovery drills.

How to Apply

Please share your resume and (if available) one or more of the following:

- (a) a short summary of a database-centric or serverless transactional system you built (Aurora/PostgreSQL, stored procedures/functions, and API layers),

• (b) an example of performance tuning or reliability work you led (query plan/indexing, concurrency/locking, migration strategy, RTO/RPO drills), or
• (c) a brief write-up explaining a technical trade-off you made (e.g., correctness vs latency vs cost, logic in SQL vs application layer, serverless vs containers).