# Full Stack Mastery – Node.js, Express, SQL, PostgreSQL, MongoDB, Cloud, Redis & Production Patterns

**Ultimate master documentation combining all concepts, theory, production-ready patterns, complex queries, cloud integrations, caching, transactions, and advanced Node.js & Express practices in a single reference. Designed for senior backend engineers aiming for 7+ years mastery.**

---

## 1️⃣CORE CONCEPTS

### 1.1 Node.js

- Event-driven, asynchronous, non-blocking I/O
- Single-threaded with event loop
- Modules (`require` / `import`)
- Streams, buffers, process management, async/await

### 1.2 Express.js

- Routing & middleware
- Error handling & async error patterns
- Production structuring: MVC, feature-based folders
- Security: Helmet, CORS, rate limiting
- Logging: Winston, Pino, Sentry integration

### 1.3 DBMS Theory

- SQL vs NoSQL concepts
- Normalization / denormalization
- Transactions & ACID
- Isolation levels
- Indexing: single, compound, partial, unique, text
- Sharding, replication, caching

---

## 2️⃣DATABASES WITH COMPLEX QUERIES

### 2.1 PostgreSQL

- **Joins**: inner, left, right, full, multi-table
- **Aggregations**: COUNT, SUM, AVG, MIN, MAX, GROUP BY, HAVING
- **Window Functions**: RANK(), ROW_NUMBER(), SUM() OVER
- **Subqueries**: IN, EXISTS, correlated
- **Transactions**: BEGIN, COMMIT, ROLLBACK
- **Indexing**: B-Tree, GIN, GiST, partial indexes

- **Production Example**: Complex report with last 3 orders per user

```
SELECT u.id, u.name, SUM(o.total) AS total_spent,
JSON_AGG(o ORDER BY o.created_at DESC LIMIT 3) AS last_orders
FROM users u
JOIN orders o ON u.id = o.user_id
JOIN payments p ON o.id = p.order_id
WHERE o.status = 'PAID' AND p.status = 'COMPLETED'
GROUP BY u.id;
```

### 2.2 MongoDB / Mongoose

- **Aggregation Pipeline**: $match, $group, $lookup, $project, $slice
- **Joins with $lookup / populate**
- **Window Functions**: $setWindowFields
- **Transactions**: session-based ACID operations
- **Caching Example**: Redis for repeated queries

```
const users = await User.aggregate([
  { $lookup: { from: 'orders', localField: '_id', foreignField:
'userId', as: 'orders' } },
  { $unwind: '$orders' },
  { $match: { 'orders.status': 'PAID' } },
  { $group: { _id: '$_id', name: { $first: '$name' }, totalSpent: {
$sum: '$orders.total' } } }
]);
```

### 2.3 Redis Caching

- **Key-value caching** with TTL
- **Middleware** for cache-first retrieval
- Example:

```
const cacheMiddleware = async (req, res, next) => {
  const data = await redisClient.get(req.originalUrl);
  if (data) return res.json(JSON.parse(data));
  next();
};
```

---

## 3️⃣NODE.JS & EXPRESS ADVANCED CONCEPTS

### 3.1 Middleware Patterns

- Global, router-level, error-handling
- Async middleware wrapper
- Rate limiting, authentication, validation

### 3.2 File Uploads & Cloud

- **Multer** for multipart/form-data
- **AWS S3** upload
- **Cloudinary** for images/videos

```
const result = await cloudinary.uploader.upload(file.path);
```

### 3.3 Authentication & Authorization

- JWT for stateless auth
- Role-based access control
- Refresh tokens & revocation
- Passport.js strategies

### 3.4 Async Patterns & Error Handling

- Async wrapper function
- Try/catch with next()
- Global error handler

```
const asyncHandler = fn => (req, res, next) => Promise.resolve(fn(req,
res, next)).catch(next);
```

### 3.5 Performance & Scaling

- PM2 for process management
- Clustering for multi-core utilization
- Compression, gzip, Brotli
- CORS, Helmet, rate limiting
- Redis caching for heavy DB queries

---

# 4️⃣PRODUCTION CASE STUDIES

### 4.1 E-commerce Feed API

- Cursor-based pagination
- Aggregated orders + payments
- Redis caching first N pages
- Example code for Node + MongoDB aggregation included

### 4.2 Payment Transaction API

- MongoDB transactions for debits/credits
- PostgreSQL ACID transaction example
- Deduplication with unique indexes
- Error handling with rollback and logging

### 4.3 File Upload API

- Multiple file upload to S3/Cloudinary
- Save URLs in DB
- Serve files securely with signed URLs

### 4.4 Analytics Dashboard

- Aggregation pipelines (MongoDB)
- Weekly/monthly summaries
- Store heavy aggregates in Redis
- Serve fast dashboards

---

## 5️⃣FULL STACK MENTAL MODEL

- **Node.js**: async engine for backend logic
- **Express.js**: routing & middleware orchestration
- **DBMS**: data integrity & advanced querying
- **Cloud Storage**: offload static/large files
- **Redis**: caching & performance optimization
- **Security & Auth**: JWT, RBAC, input validation, helmet, rate limiting
- **Production patterns**: logging, monitoring, error handling, scaling

---

📌 **This document combines: theory + complex SQL/PostgreSQL/MongoDB queries + Node.js & Express production-ready code + cloud storage + caching + transactions + advanced patterns**. It serves as **a single reference for full-stack senior mastery.**