Professional Root Cause Analysis (RCA) Template specifically tailored for **SQL Server & Database Administration issues** that occur in real-time environments. You can reuse it for outages, performance incidents, data issues, job failures, etc.

SQL Server / DBA Incident RCA (Root Cause Analysis & Fix) Template

1. Incident Summary
Incident Title:
Date & Time of Incident:
Reported By:
Environment: (Prod / UAT / Dev / DR)
Impact Summary:

- (e.g., Application downtime, slow queries, failed ETL, blocking, data corruption risks, etc.)

2. Incident Timeline

Time (UTC/local)	Event Description
HH:MM	Issue detected (monitoring alert/user report)
HH:MM	DBA team engaged
HH:MM	Initial investigation started
HH:MM	Root cause identified
HH:MM	Fix applied
HH:MM	System validated & normalized
HH:MM	Incident closed

3. Detailed Incident Description
Provide a clear description of what happened:

- What SQL Server component was affected?
  (Engine, Query Performance, TempDB, Agent Job, AlwaysOn, Replication, Storage, etc.)
- What symptoms were observed?
  (High CPU, blocking, I/O latency, query timeout, job failure)

4. Root Cause Analysis
Break down the investigation using the 5 Whys or RCA approach:
4.1 Technical Root Cause

- Primary cause of the incident:
  (Examples)
  - Long-running query performing table scan due to missing index
  - TempDB contention
  - Transaction log full
  - SQL Server memory pressure
  - Deadlock between Stored Procedures X and Y
  - AlwaysOn Availability Group failover due to network packet drops
  - Disk latency spikes on data/log volumes

4.2 Contributing Factors

- Example contributing factors:
  - High workload at peak time
  - Outdated statistics
  - Incorrect execution plan due to parameter sniffing
  - SQL Agent job misconfiguration

- Lack of index maintenance
- Storage subsystem issue

5. Evidence Collected

Attach logs or reference data:

- SQL Server Error Log excerpts
- Extended Events / Profiler trace excerpts
- sp_whoisactive / sys.dm_exec_requests output
- Wait statistics
- Blocking/Deadlock graphs
- Performance Monitor metrics
- Storage/Network metrics

6. Impact Analysis

- Affected Applications/Systems:
- Number of Users Impacted:
- Duration of Impact:
- Business Impact:
  - e.g., delays in order processing, failed financial batch, unavailability of reporting system

7. Immediate Fix Applied

Describe what was done to resolve the issue **in real time**:

Examples:

- Killed blocking SPID
- Rebuilt/Reorganized index
- Cleared TempDB contention by increasing data files
- Increased log file size
- Forced failover of AlwaysOn AG
- Updated statistics
- Restarted SQL Server service (if absolutely required)
- Corrected SQL Agent job configuration

8. Permanent Preventive Actions

List long-term fixes to avoid recurrence:

Technical Actions

- Create missing indexes
- Update code to avoid table scan / RBAR operations
- Adjust MAXDOP / Cost Threshold settings
- Implement alerts for log file usage
- Add more TempDB files
- Redesign Stored Procedure to avoid deadlocks
- Patch SQL Server to latest CU
- Implement query/store plan forcing

Process Actions

- Review and optimize deployment process
- Improve monitoring thresholds
- Add capacity planning review
- Document maintenance tasks

9. Lessons Learned

- What went well during resolution?
- What could be improved?
- Were monitoring/alerts adequate?
- Could the issue have been prevented earlier?

10. Approvals

Name	Role	Approval
DBA Lead		✓
Application Owner		✓
IT Manager		✓

SQL Server / DBA Incident RCA – Example (Blocking Issue)

1. Incident Summary

Incident Title: Severe Blocking Due to Missing Index
Date & Time: 2025-11-18 14:10 EST
Reported By: Application Support Team
Environment: Production
Impact Summary:
Application experienced response time degradation.
Multiple user transactions were stuck due to a long-running report query causing blocking across OLTP workload.

2. Incident Timeline

14:10 – Monitoring alert reported high blocking chain
14:12 – DBA team engaged
14:13 – Investigation started using sp_whoisactive
14:15 – Root cause identified (report query missing index)
14:18 – Fix applied (created missing index)
14:21 – System normalized, blocking cleared
14:25 – Incident closed

3**. Detailed Incident Description**
A reporting query executed by the analytics application performed a full table scan on SalesOrderHistory (180M rows).
This caused a head blocker, impacting more than 60 concurrent OLTP transactions.

4. Root Cause Analysis
Primary Root Cause:
• A long-running query triggered a table scan due to a missing composite index on (CustomerID, OrderDate).

Contributing Factors:
• Statistics were outdated
• Report scheduled during peak business hours
• High concurrent insert/update workload

5. Evidence Collected
• sp_whoisactive output showing head blocker (SPID 91)
• sys.dm_exec_query_stats showing high logical reads
• Wait types recorded: LCK_M_S, LCK_M_X
• SQL Server Error Log verified no failovers

6. Impact Analysis

Applications Affected: Order Management, CRM
Users Impacted: ~350 concurrent users
Duration: 11 minutes
Business Impact: Slow order placement, delayed customer responses

7. Immediate Fix Applied

• Identified missing index and created the following:

```
CREATE INDEX IX_SalesOrderHistory_Cust_OrderDate
ON SalesOrderHistory(CustomerID, OrderDate);
```

• Cleared blocking automatically once index was created

8. Permanent Preventive Actions

• Add the missing index to DEV, UAT, DR environments
• Move reporting workload to read replica
• Update statistics nightly instead of weekly
• Implement blocking alerts < 30 seconds threshold

9. Lessons Learned

• Monitoring worked well; blocking quickly detected
• Need stricter control on heavy report execution timing
• Statistics maintenance frequency must be improved

10. Approvals

DBA Lead – ✓
Application Owner – ✓
IT Manager – ✓