DP-300 Case Study – Monitoring & Query Optimization

Participant Case with Inline Trainer Answer Notes

How to use this document (Trainer Guidance)
- Let participants read each problem section first.
- Ask them to propose solutions verbally or on a whiteboard.
- Then reveal the **Trainer Answer Notes** provided immediately after each problem.

---

1. Organization Context

ContosoPay Services is a digital payments platform processing UPI, card, and wallet transactions across India.

- Average Transactions per Day: ~6 million
- Peak Transactions per Hour: ~900,000
- Data Type: Financial transactions + customer identifiers
- Availability Requirement: Very High

The core transactional workload runs on Azure SQL.

---

2. Azure SQL Environment

- Azure SQL Database (Single Database)
- Service Tier: General Purpose
- Compute Model: Provisioned
- vCores: 8
- Storage: 2 TB
- Zone Redundancy: Disabled

The database supports OLTP transactions, reporting dashboards, and fraud-detection queries.

---

3. Business Expectations

- Consistent performance during peak load
- Predictable scaling behavior
- Proactive detection of issues
- Queries must meet strict SLA requirements

---

4. Situation After Go-Live

Within one month of production usage, the operations team reports frequent slowdowns during peak hours.

You are assigned as the **Azure SQL Administrator** to investigate.

PROBLEM AREA 1 – Monitoring Operational Resources

Observed Symptoms

- CPU usage sustained at **85–95%** during peak hours
- DTU/vCore pressure is continuous, not spiky
- Memory utilization remains high
- Storage I/O latency increases during reporting windows

❓ Participant Challenge

How would you **monitor and analyze operational resources** to understand what is causing sustained pressure?

✅Trainer Answer Notes

Participants should identify and use:

- **Azure Monitor Metrics**:
- CPU percentage
- Data IO percentage
- Log IO percentage
- Memory usage
- Configure **alerts based on sustained thresholds**, not momentary spikes
- Correlate metrics with workload timing (OLTP vs reporting)
- Establish performance baselines using historical metric trends

Key DP-300 Concepts: - Azure Monitor - Metrics vs logs - Proactive alerting

---

PROBLEM AREA 2 – Diagnosing Resource Bottlenecks

Observed Symptoms

- CPU remains high even when transaction volume stabilizes

- Storage latency spikes during reporting workloads
- Performance does not recover immediately after peak hours

❓ Participant Challenge

How would you **determine the actual bottleneck** instead of guessing or immediately scaling resources?

✅Trainer Answer Notes

Expected reasoning:

- Sustained high CPU indicates inefficient queries or missing indexes
- Storage I/O contention suggests competing read/write workloads
- Memory pressure can cause plan instability and longer execution times

Expected tools:

- `sys.dm_db_resource_stats`
- Azure Monitor trend charts
- Query Store runtime statistics

Key DP-300 Concepts: - Resource diagnosis - Correlating metrics with database behavior

---

PROBLEM AREA 3 – Query Performance Degradation

Observed Symptoms

- Queries that previously ran in milliseconds now take seconds
- Same query shows inconsistent execution times
- Read-heavy queries block write operations
- Long-running SELECT statements affect transaction throughput

❓ Participant Challenge

How would you **identify and optimize problematic queries** without changing application code?

✅Trainer Answer Notes

Expected approach:

- Enable and use **Query Store** to identify:
- Top CPU-consuming queries
- Long-duration queries
- Detect **execution plan regression** and force stable plans if required

- Analyze blocking patterns caused by read/write contention
- Apply indexing strategies (including automatic indexing)

Key DP-300 Concepts: - Query Store - Execution plans - Blocking and concurrency

---

PROBLEM AREA 4 – Inconsistent Query Performance

Observed Symptoms

- Query performance varies significantly between executions
- No deployment changes were made during slowdowns

❓ Participant Challenge

What could cause this behavior in Azure SQL, and how would you stabilize performance?

✅Trainer Answer Notes

Expected reasoning:

- Plan cache changes due to parameter sensitivity
- Resource pressure affecting plan selection
- Query plan regression detected via Query Store

Stabilization techniques:

- Use Query Store to identify regressed plans
- Force last known good plan
- Address root causes such as missing indexes or skewed statistics

Key DP-300 Concepts: - Plan regression - Query Store plan forcing

---

PROBLEM AREA 5 – Continuous Optimization Strategy

Observed Symptoms

- Manual tuning is reactive and inconsistent
- No long-term optimization strategy exists

❓ Participant Challenge

How would you design a **continuous monitoring and optimization approach** aligned with Azure SQL best practices?

✅Trainer Answer Notes

Expected strategy:

- Enable **Automatic Tuning** (create/drop index)
- Regularly review Query Store insights
- Use Azure SQL Insights or Log Analytics for centralized monitoring
- Scale compute only after query optimization

Key DP-300 Concepts: - Intelligent performance - Automatic tuning - Scale vs optimize decision-making

---

Final Trainer Wrap-Up Notes

Common DP-300 exam traps to highlight:

- Scaling resources without fixing inefficient queries
- Treating monitoring and troubleshooting as the same activity
- Ignoring sustained pressure trends
- Not using Query Store as the primary tuning tool

End of Case Study with Inline Trainer Notes