DP-300 Case Study – Advanced Monitoring & Query Optimization

Participant Case Study with Inline Trainer Answer Notes

Trainer Usage Guidance
This case study is intentionally **complex and slightly confusing**, mirroring real enterprise Azure SQL environments.
Allow participants to struggle, debate, and even take wrong paths before revealing the **Trainer Answer Notes** placed after each problem.

---

1. Organization Context

Aquila Logistics Systems (ALS) is a large supply-chain and logistics platform supporting:

- Real-time shipment tracking
- Warehouse inventory updates
- Partner-facing analytics dashboards

ALS operates across multiple regions in India and Southeast Asia.

- Average Daily Transactions: ~9 million
- Peak Concurrency: Extremely high during end-of-day reconciliation
- Data Pattern: Heavy writes + unpredictable read spikes

---

2. Azure SQL Environment (Intentionally Mixed)

- Azure SQL Database (Elastic Pool)
- Pool Service Tier: General Purpose
- Total Pool vCores: 16
- Databases in Pool: 6
- One primary OLTP database consumes ~60% of workload
- Zone Redundancy: Enabled

Workloads: - Continuous OLTP writes - Scheduled analytics queries every 30 minutes - Ad-hoc reporting by operations team

---

3. Business Expectations (Conflicting by Design)

- No noticeable performance degradation for OLTP workloads
- Analytics queries must complete within 5 minutes
- Infrastructure cost must remain stable

- Platform should auto-heal without frequent manual intervention

---

4. Situation After Go-Live

After 6 weeks in production, performance complaints start coming from **different teams with contradictory observations**.

You are assigned as the **Senior Azure SQL Administrator**.

PROBLEM AREA 1 – Confusing Resource Signals

Observed Symptoms

- Elastic pool CPU averages at only **55–60%**
- Individual database CPU intermittently hits **95%**
- DTU/vCore alerts fire inconsistently
- Some dashboards show no issues while applications time out

❓ Participant Challenge

How would you correctly **interpret resource utilization** in an elastic pool environment?

✅Trainer Answer Notes

Expected reasoning:

- Pool-level metrics hide **per-database saturation**
- Individual database CPU spikes can still cause throttling
- Elastic pools require **both pool and database-level monitoring**

Correct monitoring approach:

- Azure Monitor metrics at:
- Elastic pool level
- Individual database level
- Identify noisy-neighbor behavior

Key DP-300 Concepts: - Elastic pool resource governance - Pool vs database metrics

---

PROBLEM AREA 2 – Inconsistent Query Performance Across Databases

Observed Symptoms

- Same reporting query runs fast in the morning but slow at night
- Query Store shows multiple execution plans
- No code or index changes during slow periods

❓ Participant Challenge

Why does the **same query behave differently**, and how would you stabilize it?

✅Trainer Answer Notes

Expected reasoning:

- Parameter-sensitive plans due to varying data distribution
- Resource contention inside the elastic pool
- Plan regression detected via Query Store

Stabilization approach:

- Use Query Store to:
- Compare plans
- Identify regressed plans
- Force stable plans if needed
- Address resource contention before scaling

Key DP-300 Concepts: - Query Store - Plan regression - Parameter sensitivity

---

PROBLEM AREA 3 – False Scaling Assumptions

Observed Symptoms

- Team scales elastic pool to 24 vCores
- Performance improves briefly, then degrades again
- Costs increase significantly

❓ Participant Challenge

Why did scaling **not permanently solve** the issue, and what should have been done first?

✅Trainer Answer Notes

Expected reasoning:

- Scaling masks inefficient queries temporarily
- Root cause remains unoptimized workloads
- Elastic pools amplify inefficient resource usage

Correct approach:

- Identify top resource-consuming queries
- Optimize indexes and execution plans
- Scale only after workload optimization

Key DP-300 Concepts: - Scale vs optimize - Cost-performance balance

---

PROBLEM AREA 4 – Blocking Without Obvious Locks

Observed Symptoms

- Blocking occurs even with READ COMMITTED isolation
- No long-running transactions visible
- Blocking appears only during analytics execution windows

❓Participant Challenge

What could cause **hidden blocking**, and how would you confirm it?

✅Trainer Answer Notes

Expected reasoning:

- Read queries competing for resources with write-heavy OLTP
- Index scans causing resource starvation
- Lock escalation under pressure

Confirmation tools:

- Query Store wait statistics
- Blocking and wait-type analysis

Key DP-300 Concepts: - Concurrency - Wait statistics - Resource contention vs locks

---

PROBLEM AREA 5 – Monitoring Blind Spots

Observed Symptoms

- Alerts fire after users complain
- No correlation between metrics and incidents
- No historical performance context

❓ Participant Challenge

How would you redesign monitoring to **predict issues instead of reacting**?

✅Trainer Answer Notes

Expected strategy:

- Create alerts based on **sustained thresholds**
- Use Azure SQL Insights for centralized analysis
- Establish baselines for pool and databases
- Correlate metrics with Query Store insights

Key DP-300 Concepts: - Proactive monitoring - Baseline-driven alerts

---

PROBLEM AREA 6 – Long-Term Optimization Strategy

Observed Symptoms

- Manual tuning does not scale
- DBA team overloaded

❓ Participant Challenge

How would you implement a **self-optimizing Azure SQL strategy**?

✅Trainer Answer Notes

Expected approach:

- Enable Automatic Tuning (index create/drop)
- Continuous Query Store review
- Periodic workload evaluation
- Scale pools strategically

Key DP-300 Concepts: - Intelligent performance - Automatic tuning

Final Trainer Emphasis – Why This Case Is Tricky

Common confusion points:

- Pool metrics hiding database-level issues
- Mistaking low average CPU for healthy performance
- Scaling before optimization
- Confusing blocking with locking

DP-300 Exam Alignment Summary

This case reinforces:

- Monitoring elastic pools correctly
- Diagnosing misleading metrics
- Query Store as the primary optimization tool
- Cost-aware scaling decisions

End of Advanced Case Study