DP-300 Complete Notes Outline (2025 Edition)

1. Introduction to the DP-300 Exam

✓ Overview of Microsoft DP-300

The **DP-300: Administering Relational Databases on Microsoft Azure** exam is part of the Microsoft Certified: **Azure Database Administrator Associate** certification.
It focuses on managing and optimizing modern database solutions in Azure, including **deployment**, **monitoring**, **security**, **automation**, and **performance tuning**.

It tests your ability to work with **Azure SQL**, **SQL Server on Azure VMs**, and **Azure Database for PostgreSQL/MySQL**.

✓ Who Should Take the Exam?

This exam is ideal for:

- **Database administrators** with experience in on-prem and cloud environments.
- **Data professionals** who manage data platforms.
- **IT professionals** transitioning to **cloud-based database roles**.
- **SQL developers or system admins** looking to build Azure database expertise.

Pre-requisites (recommended, not mandatory):

- Experience with **SQL Server** and **basic Azure services**.
- Familiarity with **PowerShell**, **Azure CLI**, and **T-SQL scripting**.

✓ Career Benefits and Salary Impact

Certification benefits include:

- Higher **job eligibility** for roles like **Azure DBA**, **Cloud Data Engineer**, and **Database Reliability Engineer**.
- Stronger **professional credibility** in cloud database management.
- Improved **salary prospects** – certified Azure DBAs often command **$90,000 to $130,000+** in the U.S. or similar high-paying roles globally.

According to industry reports:

- Certifications like DP-300 can increase salary by **15–20%**.
- Cloud database skills are in **high demand** across enterprises.

✓ Exam Structure and Domains

As of the current exam outline, the DP-300 includes these main domains:

Domain	Weight
Plan and Implement Data Platform Resources	15–20%
Implement a Secure Environment	15–20%
Monitor and Optimize Operational Resources	15–20%
Optimize Query Performance	5–10%
Perform Automation of Tasks	10–15%
Plan and Implement a High Availability and Disaster Recovery (HADR) Environment	15–20%
Perform Administration by Using T-SQL	10–15%

Format:

- 40–60 questions
- Types: Multiple choice, drag-and-drop, case studies, scenario-based
- Duration: 100–120 minutes
- Passing score: 700/1000

✓ RealExamCollection Dumps as Study Support

RealExamCollection and similar websites provide **exam dumps** (collections of real or simulated past questions). However, there are **critical concerns**:

Pros (if used cautiously):

- Exposure to question **patterns** and **difficulty level**.
- Helpful as a **review tool** to test readiness (after deep study).

Cons and Warnings:

- Dumps may violate **Microsoft's exam policies**.

- They often contain **inaccurate or outdated** questions.
- Over-reliance can lead to **poor conceptual understanding**, risking job performance.
- Microsoft may **invalidate certifications** obtained through improper preparation.

Better alternatives:

- Microsoft Learn (official modules)
- Whizlabs, MeasureUp, or A Cloud Guru
- Books like "Exam Ref DP-300"
- Practice labs (e.g., Azure Sandbox, GitHub resources)

Summary

The **DP-300** is a highly valuable exam for anyone moving into or advancing in **cloud database administration**. It requires a balance of **practical Azure skills**, **SQL expertise**, and a **strong study plan** using official content. Use practice tools and questions ethically—**real skill beats memorization** every time.

2. Azure Database Fundamentals

✓ PaaS vs IaaS vs SaaS

These are **cloud service models**, each offering different levels of control, flexibility, and management:

Model	Full Form	What It Means	Azure Example
IaaS	Infrastructure as a Service	You manage the OS, DB, and apps. Azure provides VMs, networking, and storage.	**Azure Virtual Machines (SQL on VM)**
PaaS	Platform as a Service	Azure manages OS, DB engine, patches, and backups. You focus on databases and apps.	**Azure SQL Database**, **Azure SQL Managed Instance**
SaaS	Software as a Service	Everything (infrastructure + software) is managed. You just use the app.	**Microsoft 365**, **Power BI**

Summary:

- **IaaS = full control**, more maintenance.
- **PaaS = less admin**, more agility.
- **SaaS = no admin**, just use it.

✓ Azure SQL Database Tiers

Azure SQL Database (PaaS) offers **service tiers** for different workloads, with varying performance, pricing, and features.

Main tiers:

1. **Basic / General Purpose** – For light workloads, dev/test. Cheaper.
2. **Standard / Business Critical** – Higher availability, IOPS, and speed.
3. **Premium** – Low-latency and high-performance databases.
4. **Hyperscale** – For massive databases (up to 100 TB), instant scaling.

You choose tiers based on:

- **Performance** needs (e.g., high transactions)
- **Redundancy & failover**
- **Cost vs features**

✓ DTU vs vCore Models

These are **pricing and performance models** in Azure SQL Database.

1. DTU (Database Transaction Unit) model:

- **Pre-configured bundles** of CPU, memory, and I/O.
- Simple to use but less flexible.
- Best for small apps or when you don't want to fine-tune resources.

💡 Example: 50 DTUs = enough power for a small app with light workloads.

2. vCore (Virtual Core) model:

- You choose **CPU (vCores)**, **RAM**, and **storage** separately.
- More transparent and **cost-effective** for large/production workloads.
- Allows **Azure Hybrid Benefit** (if you already have SQL Server licenses).

💡 Example: 4 vCores, 16 GB RAM, 250 GB storage = medium workload flexibility.

📌 **DTU is easier, vCore is more customizable**.

✓ SQL Managed Instance vs Azure SQL VM

Both are ways to run **SQL Server in Azure**, but they differ in how much you manage:

Feature	Azure SQL Managed Instance	Azure SQL VM
Service Model	**PaaS**	**IaaS**
Management	Microsoft manages OS, updates, backups	You manage everything (like on-prem)
Features	Near full SQL Server compatibility (agent, cross-db queries)	100% SQL Server control
Use Case	Lift-and-shift with minimal admin	Full SQL Server control, custom settings
Pricing	Higher due to automation and PaaS benefits	Can be cheaper with licensing (Hybrid Benefit)

🔹 Use **Managed Instance** if you want **low admin overhead** with advanced SQL features.

🔹 Use **SQL on VM** if you need **legacy setup**, full control, or unsupported features in PaaS.

Summary

Concept	Key Point
PaaS vs IaaS vs SaaS	Levels of management responsibility (PaaS is ideal for DBAs)
SQL Database Tiers	Choose tier based on performance and size requirements
DTU vs vCore	DTU = simple bundles; vCore = customizable, transparent billing
Managed Instance vs SQL VM	Managed Instance = modern PaaS with SQL features; VM = legacy/full control

3. Deploy and Configure Azure SQL Resources

✓ Creating SQL Database via Portal, CLI, PowerShell

There are **3 main ways** to create an Azure SQL Database, depending on your preference or automation need:

1. Azure Portal (GUI)

- Easiest method, browser-based.
- Walkthrough wizard lets you choose server, performance tier, backup options.
- Good for beginners or small setups.

💡 *Example:* Go to "Create Resource" → "SQL Database" → fill form → click **Create**.

2. Azure CLI (Command-line)

- Scriptable and great for automation.
- Cross-platform (`az sql db create`)
- Ideal for DevOps and scripting in CI/CD.

💡 Example command:

```
bash
CopyEdit
az sql db create --name mydb --server myserver --resource-group mygroup --
service-objective S0
```

3. PowerShell

- Preferred in **Windows environments** or existing PowerShell automation.
- Full control over Azure resource provisioning.

💡 Example:

```
powershell
CopyEdit
New-AzSqlDatabase -ResourceGroupName "mygroup" -ServerName "myserver" -
DatabaseName "mydb" -RequestedServiceObjectiveName "S0"
```

✓ Geo-replication and Failover Groups

These features provide **business continuity** and **disaster recovery** for Azure SQL Databases:

1. Geo-replication:

- Creates a **read-only** secondary DB in another region.
- Manual failover.
- Use case: **reporting**, **read-scale**, or **disaster recovery**.

💡 Enabled on **Premium / Business Critical / Hyperscale tiers**.

2. Failover Groups:

- Provides **automatic failover** between primary and secondary databases across regions.
- Supports **read-write and read-only listener endpoints**.
- Use case: **high availability** + **automatic region failover**.

💡 Ideal for **multi-region apps** with low downtime tolerance.

✅ Elastic Pools vs Single Databases

These are **cost management models** for Azure SQL databases:

1. Single Database:

- Isolated resources (CPU, memory).
- You pay **per database**.
- Best for **steady, predictable workloads**.

💡 Ideal if each DB has **consistent usage**.

2. Elastic Pool:

- Shared resources among multiple DBs.
- You pay for the **pool**, not each database.
- Ideal for **variable, bursty workloads**.

💡 Example: 20 small databases, each active at different times.

📌 Summary:

- Use **Single DB** for consistent workloads.
- Use **Elastic Pools** for cost efficiency with multiple fluctuating workloads.

✅ Deployment Automation (ARM Templates, Bicep)

These tools automate resource provisioning using **Infrastructure as Code (IaC)**:

1. ARM Templates:

- JSON-based templates.
- Define full infrastructure: SQL DB, server, network, etc.
- Used in **CI/CD pipelines** and **repeatable deployments**.

💡 Example structure:

```
json
CopyEdit
{
  "type": "Microsoft.Sql/servers/databases",
  "name": "[parameters('databaseName')]",
  "properties": {
    "collation": "SQL_Latin1_General_CP1_CI_AS",
    "maxSizeBytes": "1073741824",
    "requestedServiceObjectiveName": "S0"
  }
}
```

2. Bicep (Newer and easier):

- A **simplified syntax** for ARM.
- Compiles into ARM under the hood.
- Easier to read and write.

💡 Bicep vs ARM: Think of **Bicep as TypeScript**, **ARM as JavaScript**.

📌 Both are used to automate SQL DB creation, backups, failover groups, etc., across environments (dev, staging, prod).

Summary Table

Topic	Summary
Creating SQL DB	Use Portal for ease, CLI/PowerShell for automation
Geo-replication & Failover Groups	For regional redundancy; failover groups = auto switch
Elastic Pools vs Single DBs	Elastic = cost-efficient for many DBs with varying loads
Deployment Automation	Use ARM or Bicep for reliable, repeatable, version-controlled deployments

4. Configure High Availability and Disaster Recovery

✓ HA/DR Design Strategies

HA/DR (High Availability and Disaster Recovery) strategies ensure business continuity and data protection.

Design goals:

- **HA** = Keep the service online during hardware or software failures (minimal downtime).
- **DR** = Recover after catastrophic failures (e.g., region-wide outages).

Azure SQL Database built-in options:

- **Zone-redundant deployments** (Business Critical tier or Hyperscale)
- **Auto-failover groups** (DR across regions)
- **Geo-replication** (manual failover)
- **Point-in-time restore** (PITR)
- **Long-term retention (LTR)** for backups

📌 Choose HA/DR tools based on **RPO** (Recovery Point Objective) and **RTO** (Recovery Time Objective) needs.

✓ Availability Zones and Sets

Availability Zones:

- **Physically separate datacenters** within the same Azure region.
- Used in **Business Critical** tier and **Hyperscale** to keep replicas in **different zones**.
- **Automatic failover** between zones if one goes down.

❗ Enabled by choosing **Zone-Redundant configuration** during deployment.

Availability Sets (for VMs):

- Ensures VM instances are distributed across **multiple fault and update domains**.
- Applies only to **IaaS-based SQL Servers (SQL on VM)**.

📌 Use **Zones for PaaS (Azure SQL DB/MI)** and **Sets for IaaS (SQL on VM)**.

✓ Failover Groups and Auto-Failover

Failover Groups provide **geo-redundancy** and **automatic failover** for Azure SQL Database and SQL Managed Instances.

Features:

- Automatic replication to **paired region** (read-write and read-only replicas).
- Use **listener endpoints** for seamless failover access (`<servername>.database.windows.net`).
- Supports **manual or automatic** failover based on health checks.

💡 Best for **mission-critical apps** needing near-zero downtime across regions.

📌 Works only with **vCore model**, not DTU.

✅ Geo-Replication Step-by-Step

Active Geo-Replication = Manual, readable secondary DB in another region (PaaS SQL DB only).

🔧 *Step-by-step:*

1. **Go to your Azure SQL database** in the Portal.
2. Click on **Geo-Replication** under "Settings."
3. Select a **target region**.
4. Choose or create a **target server** in that region.
5. Confirm and **start replication**.

💡 Secondary is **read-only** and used for reporting or DR.
💡 You can **failover manually** from portal, PowerShell, or CLI.

📌 Use Geo-replication when you need **read-scale** and **manual DR**.

✅ PITR, LTR, and Backup Policies

These are **data recovery features** for Azure SQL Database:

1. PITR (Point-In-Time Restore):

- Restore your DB to any point in the **last 7–35 days** (default = 7 days).
- Uses **automated backups**.
- Fast recovery from **accidental deletes or corruption**.

2. LTR (Long-Term Retention):

- Store full backups for **up to 10 years**.
- Useful for **audit**, **compliance**, or **archival**.

💡 Configure on each database (e.g., weekly/monthly retention policies).

3. Backup Policies:

- Azure takes **automatic backups** of your SQL DB.
- Includes **full**, **differential**, and **transaction log backups**.
- Backup frequency depends on **tier**:
  - **Every 5-10 minutes** for transaction logs
  - **Weekly full**, **daily differential**

📌 Backups are stored in **RA-GRS storage** (geo-redundant).

🔄 Summary Table

Feature	Purpose	Use Case
HA/DR Design	Ensure uptime + recovery	Always-on service and data protection
Availability Zones/Sets	Physical isolation within a region	Intra-region failover
Failover Groups	Geo-redundant auto failover	Critical apps needing RTO ~ seconds
Geo-Replication	Manual DR + read replicas	Reporting, low-cost DR
PITR	Restore recent data (7–35 days)	Accidental deletions
LTR	Store backups for years	Legal/audit/compliance
Backup Policies	Auto backups + restore support	Built-in reliability

5. Monitor and Optimize Operational Resources

✓ Azure Monitor and Log Analytics

Azure Monitor:

- **Centralized platform** for monitoring the health, performance, and availability of your Azure resources.
- Collects **metrics, logs, and diagnostics** from Azure SQL Database.
- Alerts you on issues like high DTU usage, slow queries, or failed connections.

Log Analytics (part of Azure Monitor):

- Uses **Kusto Query Language (KQL)** to query and analyze collected logs.
- Helps you **troubleshoot**, **audit**, and **visualize** SQL performance and usage patterns.

💡 You can connect Azure SQL Diagnostics to Log Analytics for **deep insights**.

✅ Query Performance Insight

A built-in feature of Azure SQL that helps you analyze **top-consuming queries** in terms of:

- **CPU time**
- **Execution count**
- **Duration**

Key benefits:

- Identify **expensive or poorly written queries**.
- Track performance **trends over time**.
- Drill into **query text**, **execution plan**, and **wait time breakdown**.

💡 Great for spotting **performance regressions** after code or schema changes.

📌 Available directly in the **Azure Portal** under your SQL DB → **Intelligent Performance** → **Query Performance Insight**.

✅ Wait Stats, Missing Indexes, CPU/Disk Bottlenecks

These are critical indicators of **SQL performance issues**:

Wait Stats:

- Show **why** SQL queries are delayed (e.g., waiting for memory, CPU, locks).
- Helps diagnose **resource contention**.

💡 Example waits: `PAGEIOLATCH` (I/O wait), `SOS_SCHEDULER_YIELD` (CPU pressure)

Missing Indexes:

- SQL Engine suggests **missing indexes** that could improve query speed.
- Can be found via DMVs (`sys.dm_db_missing_index_details`).

💡 Act only after analyzing workload impact—too many indexes can **slow writes**.

CPU/Disk Bottlenecks:

- High **CPU** = inefficient queries or underpowered tier.
- High **Disk I/O** = missing indexes, too much tempdb usage, or large reads.

📌 Monitor these via:

- **Azure Monitor Metrics**
- **Query Store**
- **Dynamic Management Views (DMVs)**

✅ Extended Events in Azure SQL

Extended Events (XEvents) are a **lightweight monitoring system** to capture and analyze:

- Long-running queries
- Deadlocks
- Errors and warnings
- Wait types, execution plans, and much more

In Azure SQL:

- You can create and manage XEvents using:
  - **T-SQL (`CREATE EVENT SESSION`)**
  - **Azure Portal (for SQL Managed Instance only)**

💡 Events are stored in **ring buffers** or **file targets** for later analysis.

📌 Best for **custom performance troubleshooting** without significant overhead.

🔄 Summary Table

Tool / Concept	Purpose	Use Case
Azure Monitor	Collects metrics/logs	Track health and usage
Log Analytics	Analyze logs with KQL	Deep diagnostics & alerts
Query Performance Insight	View top resource-consuming queries	Tune slow or high-impact queries
Wait Stats	Reveal query delays	Pinpoint CPU, I/O, locking issues
Missing Indexes	Suggest performance improvements	Speed up slow queries
CPU/Disk Monitoring	Detect bottlenecks	Optimize performance tier or code
Extended Events	Capture detailed runtime data	Advanced troubleshooting & tracing

6. Optimize Query Performance

✓ Execution Plans: Estimated vs Actual

Execution Plan:

- A **blueprint** the SQL Engine uses to retrieve query results.
- Helps DBAs and developers **understand query performance**.

Estimated Execution Plan:

- Generated **before** the query runs.
- Shows **SQL Server's prediction** of how it will execute the query.
- No runtime data—uses **statistics** and cost estimates only.

💡 Use when you want to **preview performance** without running a heavy query.

Actual Execution Plan:

- Includes **runtime data**: row counts, I/O, CPU usage, actual operations.
- Helps you **verify if the plan matched reality**.

📌 Use actual plans for **debugging slow queries** or **analyzing regressions**.

✓ Statistics and Cardinality

Statistics:

- Data about **distribution of values** in a column (e.g., how many rows per value).
- SQL Server uses them to **estimate row counts**, choose join types, and indexes.

💡 Outdated statistics = **bad estimates** = bad plans.

- Automatically updated by default, but manual updates may be needed for:
  - Large data changes
  - Poor performance

Cardinality:

- **Estimated number of rows** returned by each operation in a query.
- Based on statistics.

📌 Large gaps between **estimated vs actual cardinality** = potential **statistics issue or suboptimal plan**.

✓ Index Tuning: Clustered, Non-Clustered, Covering

1. Clustered Index:

- **Physically sorts** table rows by key column.
- Only **one per table**.
- Default index on **primary key** (unless specified otherwise).

💡 Fast for range scans (e.g., `BETWEEN`, `ORDER BY`).

2. Non-Clustered Index:

- Separate from table data.
- Points to the actual row using a **row locator** (RID or clustered key).
- Can have **multiple per table**.

💡 Ideal for **filter conditions**, **frequent searches**.

3. Covering Index:

- A **non-clustered index** that includes **all columns** needed by the query.
- Prevents need to look up the base table (no bookmark lookup).

💡 Boosts performance by reducing I/O.

📌 Use **Index DMVs** (`sys.dm_db_index_usage_stats`) + **Query Store** to tune usage.

✓ Query Store and Plan Regression

Query Store:

- Built-in feature to **track query performance** over time.
- Stores:
  - Query texts
  - Execution plans
  - Performance metrics (CPU, duration, reads)

Plan Regression:

- Happens when SQL Server **suddenly uses a worse plan** for a previously fast query.
  - Caused by plan cache eviction, parameter sniffing, or stats changes.

💡 **Query Store** helps detect and **force a good plan**:

```
sql
CopyEdit
EXEC sp_query_store_force_plan @query_id = 1, @plan_id = 2;
```

📌 Use it for **performance history, troubleshooting, and plan stabilization**.

✅ PBQ-Style Performance Tuning Examples

PBQ = Problem → Bottleneck → Query Fix

This approach helps structure performance troubleshooting:

Example 1

Problem: Query is slow with high CPU
Bottleneck: High estimated vs actual row count
Query Fix:

- Update statistics
- Rewrite query to improve cardinality
- Consider using OPTIMIZE FOR hint

Example 2

Problem: Scan on a large table
Bottleneck: Missing index
Query Fix:

- Create a non-clustered covering index

```
sql
CopyEdit
CREATE NONCLUSTERED INDEX IX_Customers_Email
ON Customers(Email)
INCLUDE (FirstName, LastName);
```

Example 3

Problem: Query slow after deployment
Bottleneck: Plan regression
Query Fix:

- Use **Query Store** to revert to previous plan
- Check parameter sniffing

Summary Table

Concept	Purpose	Key Benefit
Estimated Plan	Shows intended plan	Preview, debugging
Actual Plan	Includes runtime stats	Pinpoint real issues
Statistics & Cardinality	Improve estimate accuracy	Better plan selection
Indexes	Speed up queries	Lower I/O & CPU
Query Store	Track plans & fix regressions	Long-term visibility
PBQ Framework	Structured tuning method	Focused diagnostics

7. Implement Security

✓ Network Isolation (Firewall, VNet Rules)

Controlling **who can connect** to your Azure SQL Database is the first layer of defense.

1. Firewall Rules

- Control access **by IP address**.
- Set **server-level** (applies to all DBs) or **database-level** rules.
- Can allow or block **public internet access**.

💡 Example: Allow IP range `203.0.113.0/24` to access the DB.

2. Virtual Network (VNet) Rules

- Restrict access to specific **Azure Virtual Networks**.
- Use **Private Endpoints** to eliminate public exposure.
- Stronger isolation for **internal-only** or **secure apps**.

📌 Combine firewall + VNet for layered security.

✅ Authentication (SQL, AAD, Managed Identities)

Controlling **who you are** when accessing the database.

1. SQL Authentication

- Username/password-based.
- Traditional method.
- Must be **stored and rotated securely**.

2. Azure Active Directory (AAD) Authentication

- Use Azure AD users/groups to access the DB.
- Supports **MFA**, **SSO**, and centralized identity management.
- Preferred for **enterprise security**.

💡 Example:

```
sql
CopyEdit
CREATE USER [aad_user@domain.com] FROM EXTERNAL PROVIDER;
```

3. Managed Identities

- Use the **identity of the Azure service** (like an Azure Function or VM).
- No credentials stored—**automated, secure access**.
- Great for app-to-DB access (no hardcoded secrets).

📌 Recommended for **internal app authentication** to Azure SQL.

✅ RBAC and Least Privilege Model

Control **what authenticated users can do** in Azure.

1. RBAC (Role-Based Access Control):

- Assign roles like `Contributor`, `Reader`, or `SQL DB Contributor` at **Azure level** (resource group, DB).
- Controls who can **manage resources**, not query data.

2. SQL Roles (Inside the DB):

- Control **data-level permissions**.
- Roles: `db_datareader`, `db_owner`, `db_ddladmin`, etc.

Least Privilege Model:

- Give users **only the minimum access they need**.
- Avoid giving `db_owner` unless absolutely necessary.
- Apply **principle of least privilege** consistently.

📌 Helps reduce risk of **accidental damage or misuse**.

✅ TDE, Always Encrypted, Data Masking

These features **protect data at rest and in use**:

1. TDE (Transparent Data Encryption)

- Encrypts database, backups, and transaction logs **at rest**.
- Enabled **by default** on Azure SQL.
- No application changes needed.

💡 Uses a built-in service-managed key (or customer-managed key if needed).

2. Always Encrypted

- Encrypts **sensitive columns** (e.g., SSN, credit card) **client-side**.
- Data is encrypted **before reaching the server**.
- Only the app (with keys) can decrypt.

💡 Ensures SQL Server **never sees plaintext**.

3. Dynamic Data Masking

- Obscures sensitive data (e.g., show `XXXX-XXXX-1234` instead of full card number).
- For **read-only protection**, not encryption.

💡 Useful for **dev/test environments** or casual data access.

✅ Auditing, Threat Detection, Defender for SQL

Tools to **monitor and detect** security issues in Azure SQL:

1. Auditing

- Logs **who accessed what, when, and how**.
- Stores logs in **Log Analytics**, **Event Hub**, or **Storage Account**.
- Helps with **compliance** (e.g., HIPAA, GDPR).

💡 Example: Log all SELECT, INSERT, DELETE activity.

2. Advanced Threat Protection (Deprecated term, now part of Defender for SQL)

- Detects **suspicious activities**:
  - SQL injection attempts
  - Unusual logins
  - Data exfiltration

💡 Sends **email alerts and recommendations**.

3. Microsoft Defender for SQL

- Unified security dashboard.
- Includes **vulnerability assessment**, **threat detection**, and **security score**.
- Scans for:
  - Missing patches
  - Misconfigured access
  - Insecure permissions

📌 Recommended for **enterprise-grade SQL security** monitoring.

🔒 Summary Table

Feature	Purpose	Key Benefit
Firewall / VNet Rules	Network access control	Isolate from public internet
SQL / AAD / Managed Identity	Authentication methods	Secure user/app login
RBAC + SQL Roles	Access control	Enforce least privilege
TDE	Encryption at rest	Automatic, default protection
Always Encrypted	Encryption in use	End-to-end protection
Data Masking	Obfuscate sensitive info	Visual-only protection
Auditing	Log access & actions	Compliance, accountability

Feature	Purpose	Key Benefit
Threat Detection & Defender	Alert on threats	Proactive security response

8. Automate Database Tasks

✓ SQL Agent Jobs in Azure

In SQL Server (on-prem/VM):

- **SQL Server Agent** is used to **schedule jobs**, such as:
  - Backups
  - Index rebuilds
  - ETL workflows
  - Custom scripts (T-SQL, PowerShell)

In Azure SQL Database (PaaS):

- **SQL Agent is not available**, but you can achieve similar functionality using:

Alternative	Description
Elastic Jobs	PaaS-native job automation for Azure SQL DBs
Azure Automation + Runbooks	General-purpose automation (PowerShell, scripts)
Azure Logic Apps / Functions	For serverless job workflows
Azure Data Factory	For ETL/data movement tasks

📌 For SQL Managed Instance: **SQL Agent is supported**, just like on-prem SQL Server.

✓ Azure Automation

A **cloud-based automation platform** to run **scripts on schedule or on demand**.

Features:

- Run **PowerShell** or **Python** scripts.
- Ideal for **database maintenance tasks**, such as:
  - Index rebuilds
  - User creation
  - Auditing checks

- SQL backups for VMs

Benefits:

- **No infrastructure to manage**.
- Can authenticate using **Managed Identity**.
- Integrates with **Azure Monitor, Log Analytics, Webhooks**.

📌 Works great as a **SQL Agent alternative** for Azure SQL DB.

✓ Runbooks and Webhooks

Runbooks:

- Script files (PowerShell or Python) inside **Azure Automation**.
- You can **schedule them**, **run manually**, or **trigger by alert**.

💡 Example:

- Runbook to shrink logs or rebuild indexes weekly.

Webhooks:

- Allow external apps to **trigger a runbook via URL**.
- Secure way to call scripts **from external systems, alerts, or CI/CD pipelines**.

💡 Example: A webhook that triggers a runbook when **CPU hits 90%**, or after a GitHub deployment.

📌 Enables **event-driven automation** across systems.

✓ PowerShell for DB Admin

PowerShell is ideal for **automating Azure SQL administration** tasks, such as:

Common commands:

- **Create database**:

```
powershell
CopyEdit
New-AzSqlDatabase -ResourceGroupName "myRG" -ServerName "myServer" -
DatabaseName "myDB" -RequestedServiceObjectiveName "S0"
```

- **Set firewall rule**:

```
powershell
CopyEdit
New-AzSqlServerFirewallRule -ResourceGroupName "myRG" -ServerName "myServer"
-FirewallRuleName "AllowClient" -StartIpAddress "10.0.0.1" -EndIpAddress
"10.0.0.1"
```

- **Audit settings, TDE status, or performance tier management**.

📌 Combine with **Runbooks** for scheduling or event-driven actions.

✅ CLI Scripts for Common DBA Tasks

Azure CLI is a **cross-platform command-line tool** for automating Azure SQL tasks without needing PowerShell.

Examples:

- **Create SQL Database**:

```
bash
CopyEdit
az sql db create --name mydb --server myserver --resource-group myrg --
service-objective S0
```

- **Backup SQL VM DB (IaaS)**:
  Use CLI to trigger backup via Azure Backup, not applicable to PaaS.
- **Configure long-term backup retention**:

```
bash
CopyEdit
az sql db ltr-policy set --resource-group myrg --server myserver --database
mydb --weekly-retention P12W
```

- **Run T-SQL via CLI**:

```
bash
CopyEdit
az sql db query -n mydb -s myserver -g myrg --query-text "SELECT TOP 10 *
FROM Sales;"
```

📌 Use scripts in **CI/CD**, **Bash scripts**, or **automated workflows**.

🔄 Summary Table

Tool	Purpose	Use Case
SQL Agent (VM or MI)	Job scheduling	Automate DB tasks like backups, ETL
Azure Automation	Script automation platform	Indexing, patching, log cleanup
Runbooks	Executable PowerShell scripts	Schedule or trigger admin tasks
Webhooks	External trigger for Runbooks	Event-based automation
PowerShell	Full-featured scripting for Azure	Admin, security, monitoring
CLI Scripts	Fast automation via terminal	Cross-platform DevOps scripting

9. Perform Backup and Restore

✅ Automated Backups in Azure

Azure SQL Database (PaaS) provides **automatic, continuous backups** with **no manual setup required**.

Key Features:

- **Full backups**: Every week
- **Differential backups**: Every 12–24 hours
- **Transaction log backups**: Every 5–10 minutes

🔐 Stored in **RA-GRS** (geo-redundant storage) by default.

📌 Backup retention:

- **7 to 35 days** (configurable) for **Point-In-Time Restore (PITR)**
- Can be extended for **compliance** using Long-Term Retention (LTR)

📝 Applies to:

- Azure SQL Database (single/elastic pool)
- Azure SQL Managed Instance

✓ Point-In-Time Restore (PITR)

Allows you to **restore the database to any moment** within the **retention window** (default: last 7 days).

Use Cases:

- Accidental data deletion
- Bad deployments
- Corruption

How it works:

- Creates a **new database** from backup at a **specific time**.
- Done via **Azure Portal**, **PowerShell**, **CLI**, or **REST API**.

💡 Example via Azure CLI:

```
bash
CopyEdit
az sql db restore --dest-name newdb --name originaldb --server myserver --
resource-group myrg --time "2025-07-25T15:30:00Z"
```

📌 PITR is **fast**, but restores to a **new database**, not over the existing one.

✓ LTR (Long-Term Retention) for Compliance

Used for **audits, legal, or business compliance** where you need to store backups for **months or years**.

Key Features:

- Store **weekly full backups** for **up to 10 years**
- Can configure **weekly, monthly, yearly policies**
- Stored in **separate backup vault** (not standard PITR location)

💡 Example: "Retain 1 backup per month for 5 years"

Configuration:

- Azure Portal: Under "Manage Backups"
- Azure CLI:

```
bash
```

```
CopyEdit
az sql db ltr-policy set --name mydb --server myserver --resource-group myrg
--weekly-retention P12W --monthly-retention P5Y
```

📌 Restore LTR backups **even after the database is deleted**.

✅ Manual Backup via SSMS

Azure SQL **PaaS databases** (Azure SQL Database and MI) **do not support native manual backups** via SSMS like on-prem SQL Server.

Alternatives:

1. **Export to BACPAC file**:
   - Use **SSMS** or **Azure Portal**
   - BACPAC = schema + data (for small/mid-size DBs)
   - Stored in **Azure Blob Storage**

💡 In SSMS: Right-click DB → **Tasks → Export Data-tier Application**

2. **For SQL Server on Azure VM**:
   - Manual **.bak file backups** via SSMS **are fully supported**.

📌 Use **BACPAC** only for **archival, migration, or dev/test**—not as a true backup/restore mechanism.

✅ Restoring to New Region or Server

You can **restore a database** to:

- A **different logical server** in the same region
- A **completely different Azure region** (for DR/migration)

Options:

1. **Using PITR or LTR**:
   - Select **a different server** or **region** as the target
   - Azure handles the cross-region restore (if using geo-redundant backups)
2. **Geo-restore** (if original region is down):
   - Uses **latest geo-replicated backup**
   - Available in paired region even if original region is unavailable

💡 CLI Example:

```
bash
CopyEdit
az sql db restore --dest-name restoredDb --resource-group myrg --server
newserver --geo-redundant
```

📌 Important: Restoring to a new region takes **longer** and may have **higher RTO** depending on backup location.

🔄 Summary Table

Feature	Purpose	Notes
Automated Backups	Built-in, no setup	Used for PITR & LTR
PITR	Restore to any point in recent past	Up to 35 days
LTR	Long-term backup retention for compliance	Up to 10 years
Manual Backup (SSMS)	Only via BACPAC (PaaS)	Full .bak supported only in Azure VMs
Restore to New Region/Server	Disaster recovery or migration	Uses geo-backups

10. Configure Authentication and Authorization

✅ Role-Based Access Control (RBAC)

RBAC is used in Azure to **manage who can perform what actions on which resources**.

Key Elements:

- **Role assignments**: Bind a **user or group** to a **role** on a **scope** (e.g., subscription, resource group, server).
- Built-in roles:
  - `Reader`: View-only access
  - `Contributor`: Modify but can't manage access
  - `SQL Server Contributor`: Manage SQL servers
  - `SQL DB Contributor`: Manage SQL databases

📌 RBAC controls **resource-level access in Azure**—not actual **data access** inside the database.

💡 Use **least privilege principle**: give users only what they need.

✅ Azure AD Groups and Users

Azure Active Directory (AAD) is Azure's identity service for **centralized authentication**.

Users:

- Individual **Azure AD users** can be granted access to Azure SQL.
- Login is passwordless or integrated (SSO, MFA).

Groups:

- Azure AD **security groups** allow assigning permissions to many users at once.
- Use groups to apply **consistent and scalable access** (recommended).

Azure SQL Integration:

- Use **AAD authentication** instead of SQL auth for better security.
- Create DB users from Azure AD:

```
sql
CopyEdit
CREATE USER [alice@contoso.com] FROM EXTERNAL PROVIDER;
```

💡 Roles (like `db_datareader`, `db_owner`) can be assigned to AAD users/groups.

📌 Benefits: MFA, SSO, central identity management, no stored credentials.

✅ Login Auditing and Session Monitoring

Used to **track who accessed the database**, when, from where, and what they did.

Tools:

1. **Auditing**:

- Logs access attempts, data reads/writes, and schema changes.
- Store in **Log Analytics**, **Storage Account**, or **Event Hub**.
- Supports filtering by **action type**, **user**, etc.

💡 Example: Who executed `DELETE` statements yesterday?

2. **Diagnostic Settings**:

- Enable **SQL security audit logs** and **connection/session metrics**.

3. **Azure Monitor**:

- View sessions, failed logins, and long-running queries.

4. **Query Store**:

- Monitor query performance per user/session.

📌 Enables **compliance, forensic analysis, and anomaly detection**.

✅ Security Best Practices Checklist

A high-level **security checklist** for Azure SQL and related resources:

🔐 Access Control

- ✅ Use **Azure AD authentication**
- ✅ Assign roles using **AAD Groups**, not individuals
- ✅ Apply **RBAC** at the resource group or server level
- ✅ Follow **least privilege** principles (no `db_owner` unless necessary)

🔒 Data Protection

- ✅ Ensure **Transparent Data Encryption (TDE)** is enabled
- ✅ Use **Always Encrypted** for sensitive columns (e.g., SSNs)
- ✅ Apply **Dynamic Data Masking** for dev/test and reports

📋 Auditing & Monitoring

- ✅ Enable **Auditing** and **Log Analytics integration**
- ✅ Turn on **Microsoft Defender for SQL** (formerly ATP)
- ✅ Set up alerts for **suspicious logins** and **data exfiltration**

👤💻 Login & Credential Management

- ✅ Use **Managed Identities** for app-to-DB access
- ✅ Rotate any stored secrets with **Key Vault**
- ✅ Avoid using **SQL authentication** in production

🌐 *Network Security*

- ✅ Use **Private Endpoints** or **VNet rules**
- ✅ Disable **Allow Azure Services** unless required
- ✅ Configure **firewall rules** precisely

🔄 Summary Table

Feature	Description	Why It Matters
RBAC	Control who can manage Azure SQL resources	Prevents accidental or malicious changes
AAD Groups/Users	Centralized, secure identity	MFA, SSO, no password storage
Auditing & Monitoring	Track access & changes	Meet compliance and detect threats
Security Checklist	Best practices across all areas	Strengthens your defense posture

11. Manage Performance Using Alerts and Baselines

✅ Create Metric Alerts (CPU, DTUs)

Metric alerts help you **proactively monitor performance** and trigger actions when resource usage crosses a defined threshold.

Common Metrics to Monitor:

- **CPU usage (%)**
- **DTU usage** (if on DTU-based tier)
- **Storage size**
- **Deadlocks, failed connections, blocked sessions**

How to Set Up:

You can configure alerts in the **Azure Portal**, **Azure CLI**, or **PowerShell**.

Steps (Portal):

1. Go to your SQL Database or Server.
2. Under **Monitoring**, choose **Alerts**.
3. Click **+ New Alert Rule**.

4. Select a **signal** (e.g., "CPU Percent", "DTU Percentage").
5. Define **threshold** (e.g., CPU > 80% for 5 min).
6. Set an **action group** (email, SMS, webhook, or automation).

💡 **Best Practice**: Set alerts on both **short-term spikes** and **long-term trends**.

📌 Alerts help prevent downtime, detect anomalies, and guide scaling decisions.

✅ Set Up Query Store Baselines

Query Store tracks **query performance over time** and helps detect regressions.

Key Features:

- Stores query **plans**, **runtime stats**, and **execution history**.
- Detects when a query's **performance degrades**.
- Helps revert to a **previous faster plan**.

Setting a Baseline:

1. Enable Query Store (if not already):

```
sql
CopyEdit
ALTER DATABASE [mydb]
SET QUERY_STORE = ON;
```

2. Identify a **stable, performant period** and **force a plan**:

```
sql
CopyEdit
EXEC sp_query_store_force_plan @query_id = 123, @plan_id = 456;
```

3. Optionally set **Query Store Capture Policies**:
    - Auto vs. custom
    - Capture only long-running or frequently executed queries

💡 Use **baselines** as reference points to compare new deployments or changes.

📌 Very useful in **performance regression analysis** and troubleshooting.

✅ Thresholds for Auto-scaling and Throttling

Azure SQL offers **scaling** options based on usage, but they differ by pricing model.

🚀 *Auto-Scaling (vCore Model + Hyperscale or Serverless)*

- **Serverless Tier** can automatically:
  - **Pause during inactivity** (saves cost)
  - **Scale up/down vCores** based on usage
- You can configure:
  - **Minimum and maximum vCores**
  - **Auto-pause delay (in minutes)**

📌 Ideal for **sporadic workloads** or development environments.

🧯 *Manual Scaling (DTU or vCore)*

You can scale resources **manually** when metrics hit thresholds:

- If **DTU % > 85% consistently**, scale to higher tier.
- For **vCore model**, increase CPU/memory directly.

Use **metric alerts** to notify when thresholds are crossed:

- CPU > 80% for 15 minutes → alert
- DTU > 90% average → recommend scale-up

🚫 *Throttling*

When resource limits (CPU, DTU, IOPS) are exceeded:

- **Requests are delayed or rejected**
- Error codes like `10928` or `10929` appear

📌 Happens commonly in **DTU-based models** or under-provisioned vCore setups.

How to Mitigate:

- Monitor usage with alerts
- Use **Elastic Pools** for bursty workloads

- Scale up resources in advance or set **serverless max vCores** higher

---

🔄 Summary Table

Topic	Purpose	Key Tool
Metric Alerts	Monitor CPU, DTU, storage, etc.	Azure Monitor, Alerts
Query Store Baseline	Track and restore good plans	Query Store in SQL
Auto-scaling Thresholds	Optimize cost/performance	Serverless / Alerts
Throttling Detection	Prevent performance issues	Metrics + error codes

12. Monitor and Tune for Performance Bottlenecks

✓ TempDB Contention

TempDB is a system database used for:

- Temporary tables and variables
- Sorting operations
- Hash joins and aggregates
- Version store for row versioning

In Azure SQL:

- **You cannot configure TempDB** directly (unlike on-prem)
- However, **TempDB contention still occurs**, especially with:
  - High concurrent workloads
  - Heavy use of temp tables or sorts

Symptoms:

- Slower query performance
- Wait types like `PAGELATCH_UP` or `PAGELATCH_EX`

Mitigation:

- Reduce reliance on temp tables
- Use table variables when appropriate
- Optimize queries with sorts, joins, or window functions
- Consider **vCore models** or **Managed Instance**, which provide **more TempDB resources**

📌 In **SQL Managed Instance**, you can configure **multiple TempDB files** (like in on-prem).

✅ Resource Limits and Throttling

Azure SQL enforces **resource governance** to ensure fair use of shared infrastructure.

Common Resource Limits:

- **DTU or vCore quota**
- **Storage IOPS and throughput**
- **Concurrent worker threads**
- **TempDB usage**

Throttling Symptoms:

- Queries get delayed or fail
- Error codes like:
  - `10928`: Resource limit reached
  - `10929`: Too many concurrent requests

How to Handle Throttling:

- Monitor CPU, DTU, and IOPS using **Azure Metrics**
- Use **Elastic Pools** for bursty or multi-tenant workloads
- Scale up to a **higher SKU** or use **Serverless tier with auto-scale**
- Refactor heavy queries or batch them

💡 **Long-running or poorly indexed queries** can hit limits faster.

✅ Performance Counters via Metrics

Azure exposes many **SQL performance counters** through **Azure Monitor**:

Useful Counters:

- **CPU Percentage**: Indicates compute pressure
- **DTU Percentage**: For DTU-based models
- **Storage Used (%)**: Monitor disk space
- **Deadlocks**: Shows app/database concurrency issues
- **Blocked by other sessions**: Indicates locking contention
- **IO Requests/Sec**: IOPS usage, tied to disk limits

Access Metrics:

- Azure Portal → SQL Database → **Monitoring** → **Metrics**
- Can export to **Log Analytics** or **Grafana**

📌 You can create **alerts** and **dashboards** based on these metrics.

✅ Blocking, Deadlocks, Long-Running Queries

These are common causes of **slow performance** and **user complaints**.

🔒 *Blocking:*

Occurs when one query locks a resource and others are forced to wait.

- Often caused by:
  - Long transactions
  - Missing indexes
  - Poor query design

📌 Identify blocking chains via:

```
sql
CopyEdit
SELECT * FROM sys.dm_tran_locks
```

Or use **Azure SQL Insights**, **Query Store**, or **Extended Events**.

💥 *Deadlocks:*

Occurs when two or more queries hold locks that the other needs — results in one being killed.

- SQL Server picks a **deadlock victim** to resolve it.
- Monitor using:
  - **Extended Events**
  - **SQL Audit logs**
  - **Log Analytics** (Azure Monitor workspace)

💡 Use proper indexing, shorter transactions, and consistent access order to reduce deadlocks.

🕒 *Long-Running Queries:*

These consume resources and delay other users.

- Check in:
  - **Query Performance Insight**
  - **Query Store** (duration, CPU stats)
  - **DMVs** like `sys.dm_exec_requests`

Mitigation:

- Add indexes
- Break large operations into chunks
- Optimize query plans

🔄 Summary Table

Issue	Description	Tool/Detection
TempDB Contention	Bottleneck for temp objects	Wait stats, Managed Instance
Throttling	Resource limits cause delays	Azure Metrics, Error 10928/10929
Performance Counters	Live monitoring of usage	Azure Monitor, Metrics blade
Blocking	Locks hold up other queries	DMVs, Live Query Stats
Deadlocks	Circular locking, auto-kill	Extended Events, Logs
Long Queries	High resource use, slow apps	Query Store, Insights

13. Final Preparation Checklist

✅ Exam-Day Mindset Tips

1. **Stay Calm and Confident**
   - Trust your preparation — you've studied concepts like HA/DR, security, Query Store, and backups.
   - Expect some unfamiliar or oddly worded questions — it's normal.
2. **Be Practical, Not Perfect**
   - Focus on **real-world logic** and **Azure best practices**, not memorizing obscure syntax.
   - Think: *What would a smart DBA do in production?*
3. **Control Test Anxiety**

    - Take deep breaths before and during the exam.
    - If you're stuck, **flag the question** and move on. Don't let it drain your time or confidence.
4. **Bring a Tech-Minded Mindset**
    - Questions often relate to **troubleshooting**, **optimization**, or **design trade-offs**, not just definitions.

⚠ Commonly Asked Trick Questions

Be aware of **trap phrasing** and **misleading options**:

1. **"Least effort" vs. "Most secure"**
    - Read carefully: questions may ask for the **quickest** or **most secure** option — very different answers.
2. **"Automatically" vs. "Manually" configured**
    - Be clear on what is **enabled by default** (e.g., TDE = yes, Defender = no).
3. **"Minimize cost" vs. "Maximize performance"**
    - Pick Serverless or Elastic Pools for cost.
    - Pick Hyperscale or Business Critical for performance.
4. **Throttling vs. Autoscaling**
    - Know when scaling is **manual** vs. **automatic** (only Serverless auto-scales).
5. **Backup questions**
    - PITR is *automatic*.
    - LTR must be **manually configured**.

💡 **Tip**: Eliminate obviously wrong answers first, then choose between the last two by matching the **question goal**.

What to Skip vs. What to Master

✓ *Master These (frequently tested):*

- HA/DR strategies (Geo-replication, failover groups, PITR)
- Security (AAD auth, RBAC, TDE, auditing)
- Performance tuning (Query Store, execution plans, missing indexes)
- Monitoring (Azure Monitor, metrics, alerts)
- Deployment methods (Portal, CLI, ARM templates)

✗ *Skip or Skim:*

- Deep syntax of PowerShell or CLI unless it's **core to deployment**
- Niche features like **Geo-Fencing** or **Bicep** unless you've seen them in practice exams
- On-prem migration tools **not related to Azure** (e.g., old SSIS details)

✅ Practice Test Strategy

1. **Take Timed Mocks**
   - Use official practice tests or good-quality ones like Whizlabs or MeasureUp.
   - Aim for 80%+ consistently.
2. **Review Every Answer**
   - Understand *why* you got it wrong — not just the correct answer, but the logic behind it.
3. **Simulate Exam Environment**
   - No phones, no notes, and time pressure — simulate real conditions.
4. **Repetition is Key**
   - Revisit topics like backup types, failover tiers, identity options until they're second nature.

⏱️ Time Management During the Exam

- **Duration**: ~120 minutes
- **Questions**: Typically 40–60 questions
- **Types**: Multiple choice, case studies, drag-and-drop

Suggested Time Plan:

Question Type	Time/Question	Strategy
Standard MCQs	1.5 min	Read carefully, flag if unsure
Case Studies	10–15 min	Read questions **before** case
Drag & Drops	2 min	Think in real-world workflows

Tips:

- **Use the "flag for review"** feature wisely.
- Don't overthink early questions — keep moving.
- **Leave 10–15 minutes at the end** to review marked questions.

🎓 Bonus: Test-Day Checklist

- ✅ Good night's sleep
- ✅ Valid ID ready
- ✅ Quiet space if online (no phones, clean desk)

- ✅ Arrive/log in early (30 min buffer)
- ✅ Water bottle & quick snack if allowed

14. Practice Questions Section (50+ Scenarios)

✅ 1. PBQs (Performance-Based Questions) with Step-by-Step Logic

PBQs simulate real-life admin tasks. Below are a few examples with clear step-by-step answers.

🛠️ PBQ Example 1: Create an Azure SQL Database with Geo-Replication Enabled

Scenario:
You're tasked with deploying an Azure SQL Database in the `East US` region and enabling geo-replication to `West US` for disaster recovery.

🔍 *Steps (Portal-based):*

1. Go to **Azure Portal** > **SQL databases** > **+ Create**.
2. Choose:
    - Resource Group: `RG-Prod`
    - Database Name: `SalesDB`
    - Server: Create new or use existing in East US.
3. Pricing Tier: Choose `Standard` or `Premium` (Geo-replication requires S2 or higher).
4. Create the database.
5. After deployment, navigate to **SalesDB** > **Geo-replication**.
6. Select **West US** as the secondary region.
7. Confirm and initiate replication.

🧠 **Key Logic**:

- Geo-replication requires Premium/Business Critical or S2+ tier.
- Secondary becomes readable and failover-ready.

⚙️ PBQ Example 2: Set Up a Metric Alert When DTU Usage > 90%

Scenario:
A customer wants to be notified when DTU usage exceeds 90% for more than 10 minutes.

🔍 Steps:

1. Go to **Azure Monitor** > **Alerts** > **+ New alert rule**.
2. Scope: Select the **SQL database**.
3. Condition:
    - Signal name: `DTU Percentage`
    - Operator: `Greater than`
    - Threshold: `90`
    - Aggregation: `Average`
    - Period: `10 minutes`
4. Action Group:
    - Choose existing or create new (email, SMS, webhook).
5. Alert Rule Details:
    - Severity: Choose based on urgency (e.g., 2 = High).
    - Enable rule on creation: ✅ Yes.
6. Click **Create**.

🧠 **Key Logic**:

- Alerts help detect performance bottlenecks before throttling occurs.
- DTU model = Compute + IO + memory.

✅ 2. Case Studies with Diagrams

🗂️ Case Study: Contoso High Availability Design

Scenario:
Contoso has a mission-critical app that must remain available in the event of a regional outage. They're using Azure SQL Database (vCore, General Purpose).

Business Requirements:

- Must failover automatically
- Read-only secondary needed for reporting
- Region: Primary in `East US`, secondary in `Central US`

✅Solution Diagram:

```
sql
CopyEdit
Azure SQL DB (Primary: East US)
        |
[Auto-Failover Group]
        |
Azure SQL DB (Secondary: Central US - Readable)
```

Decision Points:

- Use **Auto-Failover Group** for:
  - **Automatic regional failover**
  - **Read-write listener & read-only endpoint**
- Choose **Business Critical** tier for better SLA and availability.

Case Study 2: Performance Troubleshooting

Scenario:
A database has slow query response times. The DBA suspects bad plans and outdated stats.

Tools:

- Query Store
- Execution Plans
- Statistics

Troubleshooting Flow:

1. Enable **Query Store** and check regressed queries.
2. Review **query plans**: Look for scans, key lookups.
3. Check **statistics**:

```
sql
CopyEdit
UPDATE STATISTICS [TableName];
```

4. Recompile or force the previous plan:

```
sql
CopyEdit
EXEC sp_query_store_force_plan @query_id = 105, @plan_id = 210;
```

🧠 **Key Logic**: Use Query Store to detect and resolve plan regressions without rewriting queries.

✓ 3. RealExamCollection Questions – Daily Drill Format

⚠️ *Note: RealExamCollection is a known braindump source. Use only for **concept drilling** after you've studied ethically with Microsoft Learn, Whizlabs, etc. Do not rely on dumps as your sole preparation.*

Drill Practice (Daily – 5 Topics x 5 Questions)

Security

1. Which encryption is enabled by default in Azure SQL Database?
2. How do you give an AAD group read access only to a specific DB?
3. What does TDE protect against?
4. What level of encryption does Always Encrypted apply to?
5. Which audit type logs queries that modify data?

Backups & Recovery

1. How many days does PITR keep backups by default?
2. How do you configure LTR in Azure Portal?
3. What is the process for restoring a deleted SQL DB?
4. Can you restore a DB to another region using PITR?
5. What is the difference between PITR and LTR?

Performance Tuning

1. Where can you view top resource-consuming queries?
2. What does a missing index warning in Query Store suggest?
3. How do you reduce key lookups in execution plans?
4. Which tool allows you to revert a bad execution plan?
5. What's the difference between clustered and covering indexes?

Deployment & Monitoring

1. What is the purpose of an ARM template?
2. How do you monitor failed logins in Azure SQL?
3. Which blade shows CPU and DTU graphs?
4. What alert type do you configure for storage nearing limit?
5. Where do you enable Advanced Threat Protection?

HA/DR

1. What's the difference between geo-replication and failover groups?
2. What is required for automatic failover between regions?
3. Can geo-secondary DBs be readable?
4. Which tier supports zone redundancy?
5. How do you promote a geo-secondary?

15. Glossary and Cheatsheets

✅ 1. Azure SQL Definitions (Quick Reference)

Term	Definition
Azure SQL Database	A PaaS-managed SQL service for single databases or elastic pools.
SQL Managed Instance	A near 100% SQL Server-compatible PaaS offering, supports SQL Agent, cross-database queries.
Azure SQL VM	IaaS model running full SQL Server in a virtual machine.
DTU (Database Transaction Unit)	Blend of CPU, memory, I/O — used in basic/standard/premium tiers.
vCore (Virtual Core)	More granular resource model; separates compute and storage, used in Gen5 and Business Critical tiers.
Elastic Pool	Group of databases sharing compute in a pool (cost-effective for variable workloads).
Failover Group	DR setup with automatic failover across regions.
Geo-Replication	Creates readable secondaries in other regions for manual failover.
Query Store	Tool that stores query plans, execution stats, and helps identify regressions.

✅ 2. Common PowerShell Commands (for Azure SQL)

🔹 Login & Connect

```
powershell
CopyEdit
Connect-AzAccount
```

🔹 Create Resource Group

```
powershell
CopyEdit
New-AzResourceGroup -Name "myRG" -Location "EastUS"
```

🔹 Create SQL Server

```
powershell
CopyEdit
New-AzSqlServer -ResourceGroupName "myRG" `
  -ServerName "myserver123" `
  -Location "EastUS" `
  -SqlAdministratorCredentials $(Get-Credential)
```

🔹 Create Azure SQL Database

```
powershell
CopyEdit
```

```
New-AzSqlDatabase -ResourceGroupName "myRG" `
  -ServerName "myserver123" `
  -DatabaseName "mydb" `
  -Edition "Standard" `
  -RequestedServiceObjectiveName "S2"
```

🔹 Enable Geo-Replication

```
powershell
CopyEdit
New-AzSqlDatabaseSecondary -ResourceGroupName "myRG" `
  -ServerName "myserver123" `
  -DatabaseName "mydb" `
  -PartnerResourceGroupName "myRG2" `
  -PartnerServerName "myserver456" `
  -AllowConnections "All"
```

🔹 Restore to Point-in-Time

```
powershell
CopyEdit
Restore-AzSqlDatabase -FromPointInTimeBackup `
  -ResourceGroupName "myRG" `
  -ServerName "myserver123" `
  -TargetDatabaseName "mydb_restore" `
  -PointInTime "2023-12-15T08:00:00Z"
```

✅ 3. High-Level Diagram References (Architecture Patterns)

🔹 PaaS HA/DR (Failover Group)

```
csharp
CopyEdit
[Azure SQL DB - East US]
        |
     Auto-Failover
        |
[Azure SQL DB - West US (readable)]
```

🔹 Elastic Pool Architecture

```
pgsql
CopyEdit
[Elastic Pool]
   ├── Database A (low usage)
   ├── Database B (bursty)
   └── Database C (medium usage)
```

🔹 SQL Managed Instance in VNet

```
css
CopyEdit
[Private VNet/Subnet]
     └── [SQL Managed Instance]
            └── NSG / Route Table
```

✅ 4. SQL Syntax Reference for Automation & Maintenance

🔹 Enable Query Store

```
sql
```

```
CopyEdit
ALTER DATABASE [SalesDB]
SET QUERY_STORE = ON;
```

🔹 Force Specific Query Plan

```
sql
CopyEdit
EXEC sp_query_store_force_plan @query_id = 13, @plan_id = 8;
```

🔹 Create SQL Agent Job (MI or VM)

```
sql
CopyEdit
USE msdb;
EXEC sp_add_job @job_name = 'BackupLogsJob';
```

🔹 Restore Database (VM only)

```
sql
CopyEdit
RESTORE DATABASE SalesDB FROM DISK = 'C:\Backups\SalesDB.bak'
WITH MOVE 'SalesDB_Data' TO 'D:\Data\SalesDB.mdf',
     MOVE 'SalesDB_Log' TO 'D:\Logs\SalesDB.ldf';
```

🔹 Monitor Long-Running Queries

```
sql
CopyEdit
SELECT * FROM sys.dm_exec_requests
WHERE status = 'running' AND cpu_time > 5000;
```

🔹 Check Active Sessions

```
sql
CopyEdit
SELECT session_id, login_name, status
FROM sys.dm_exec_sessions
WHERE is_user_process = 1;
```