Microsoft

TechCon 365 PWR CON DATA CON

Query Store and Azure SQL Copilot, who is the fairest in the land?

in /josemanueljuradodiaz

@jmJuradoDiaz

José Manuel Jurado Díaz

Jmjurado@Microsoft.com

Jose Manuel Jurado

Escalation Engineer
Microsoft

- **Services Delivery Excellence Team for PaaS databases.**
- 14 Years @ Microsoft
- **Supported SQL Server Core, Analysis Services and SQL Server On Azure VMs or other RDBMS.**
- Working with DevOps (ARM), startups companies, Developers, DBA, DBM, CIO, CFO, etc.

- **Speaker @SQL Saturday, TechReady, SQL Nexus, SQLBits, Microsoft Summit, Azure Global BootCamp, SQLKofenrenz, SQLDay.**
- Worked previously as DBA, Developer, IT Manager and other jobs for more than 25 years which I don't even remember anymore.
  Microsoft Certified Trainer and other certifications.

Juan Moreno Romo

Support Escalation Engineer
Microsoft

- **Based in Madrid, Spain**
- 9 Years @ Microsoft
- **Supporting SQL Server On-Premises**
- SQL Server Core team
- HA SME

- **Speaker @ SQL Saturday, SQL Day, Azure Global Bootcamp, Summit, NetCoreConf, CodeCamp, etc.**
- Developer, DBA, Data Modeler, Software Architect, IT Manager, Project Manager, SQL Server and SQL Server on Azure VM Support Escalation Engineer.

/JuanEMorenoRomo

@DarthSicuel

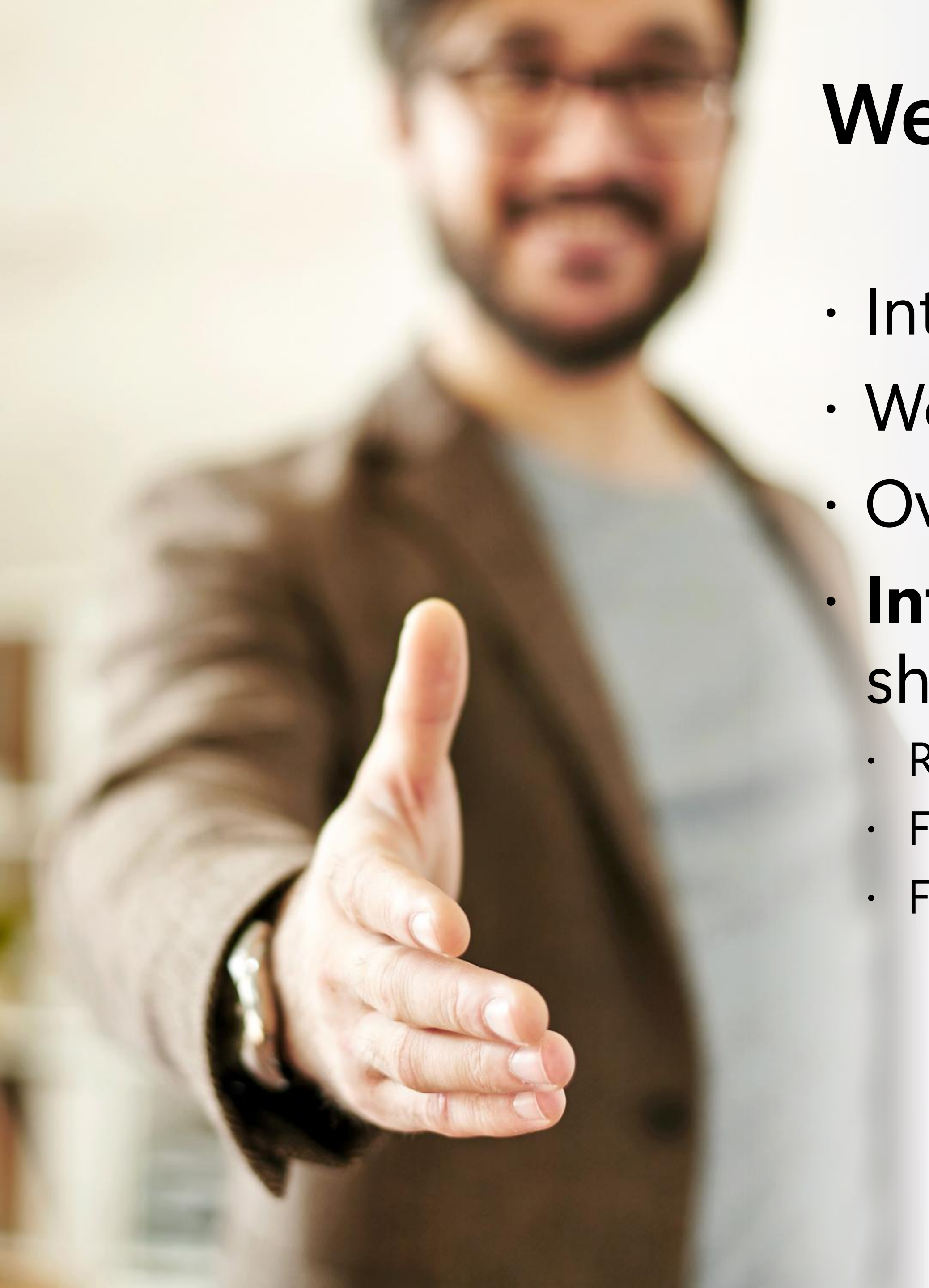
Juan Moreno Romo

jumoreno@microsoft.com

Microsoft

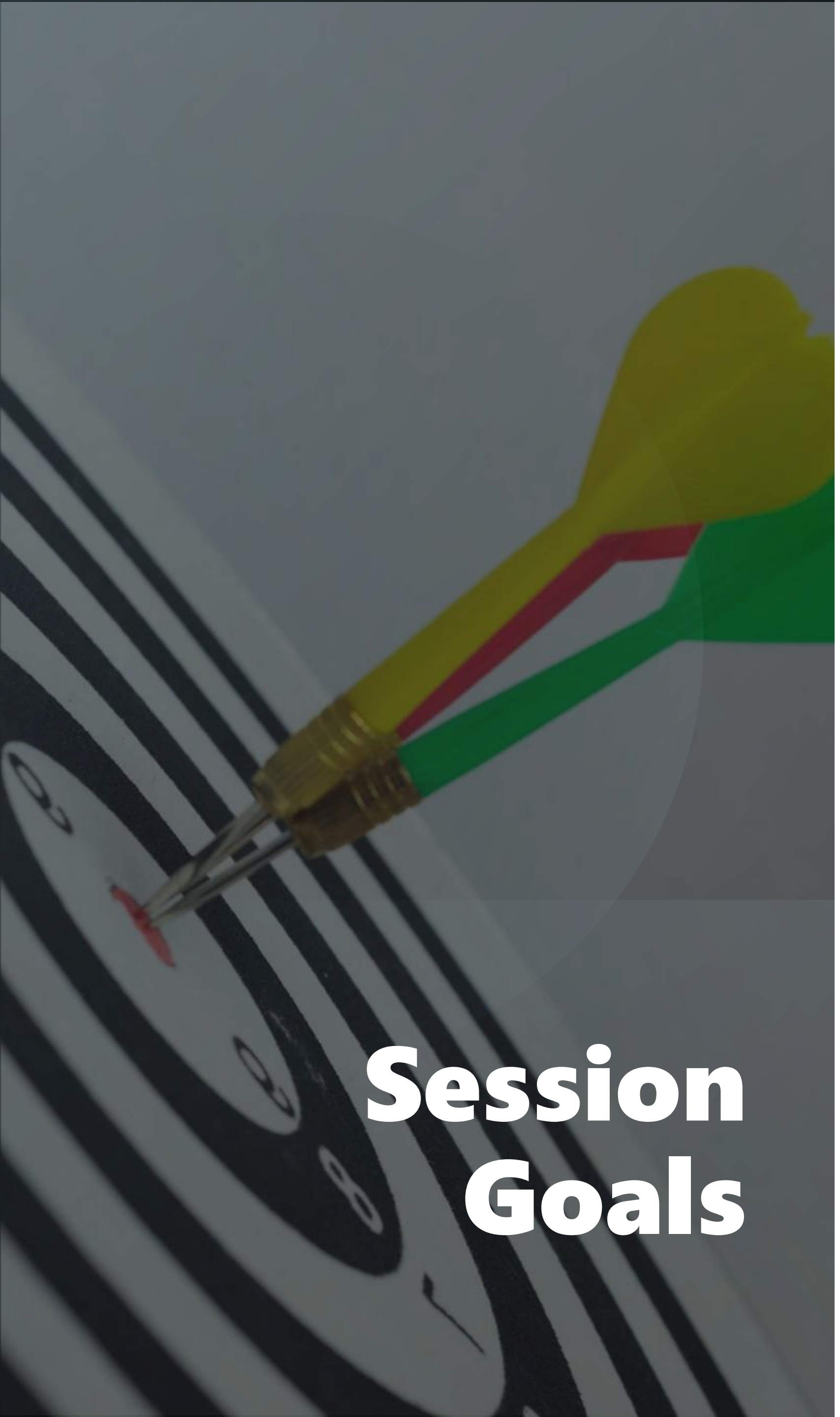
Agenda.

- Welcome & Expectations (x(15) min)
- Introduction to Query Store & Copilot (x(30) min)
- Query Store Internals & Diagnostics (x(40) min)
- Hands-on Scenarios with Python Simulation (x(90) min)
- Support Insights & Best Practices (x(20) min)
- Q&A + Wrap Up (x(15) min)



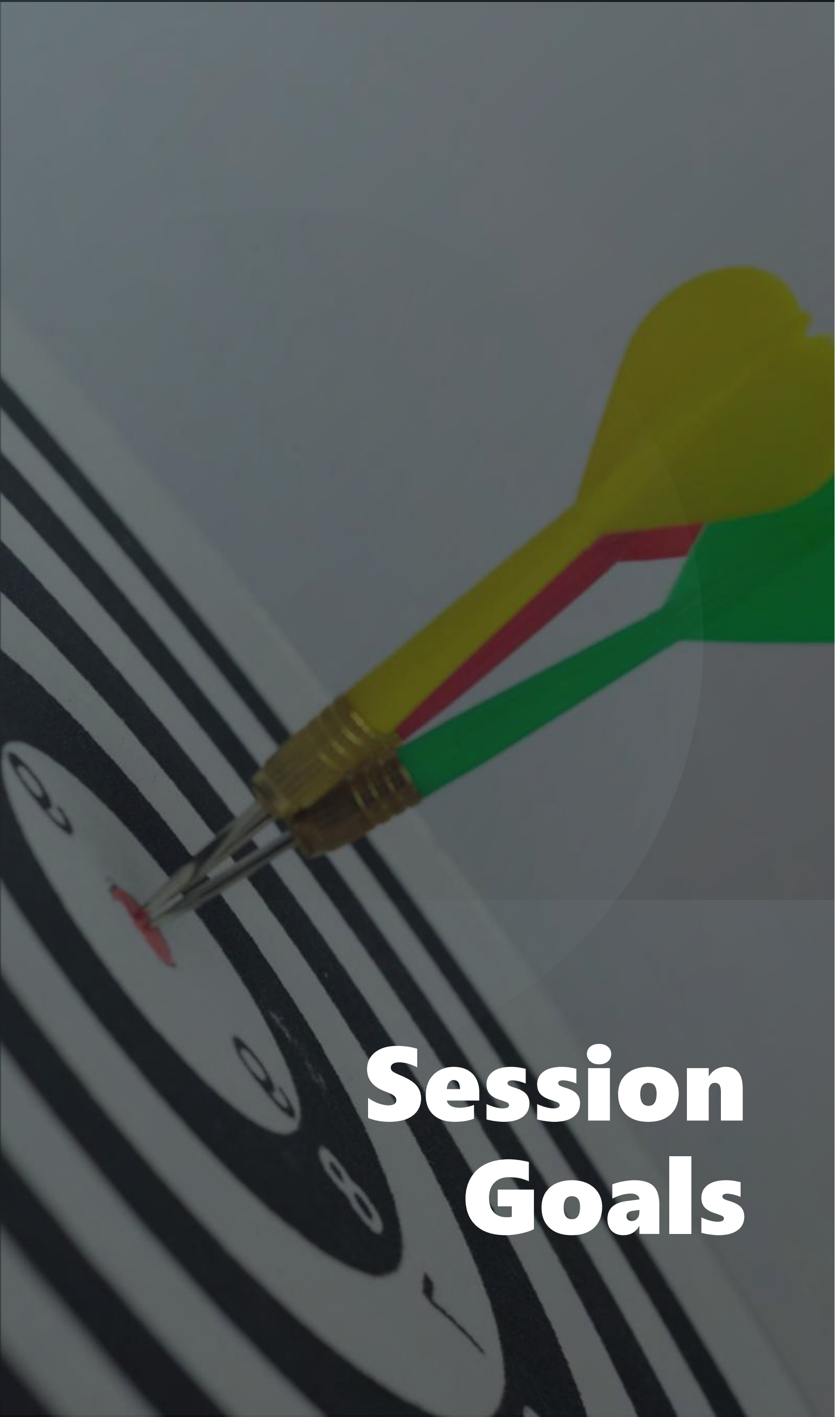
Welcome & Expectations

- Introduction of the speaker
- Workshop objectives
- Overview of the agenda
- **Interactive round:** Each participant briefly shares:
  - Role & experience with SQL Server / Azure SQL
  - Familiarity with Query Store
  - Familiarity with Copilot or AI tools



Session Goals

- Identify Common Performance Issues:
  - Recognize typical scenarios that affect application and database performance.
- Utilize Diagnostic Tools:
  - Query Data Store: Monitor queries and analyze performance patterns.
  - SSMS Copilot: Receive recommendations and solutions to detected issues.
  - Using QDS and SSMS Copilot to prevent the issues to happen
- Apply Best Coding Practices:
  - Optimize SQL queries within Python code.
  - Implement strategies to avoid locks and deadlocks.
- Foster Effective Collaboration:
  - Understand how developers and DBAs can work together to solve problems.
  - Establish communication channels to prevent future issues.



Session Goals

- Resolve Problems in Real-Time:
  - Address common errors and exceptions in Python applications connected to databases.
  - Implement retry techniques and secure transaction handling.
- Enhance Overall Application Performance:
  - Apply optimization techniques that benefit both the code and the database.
  - Comprehend the impact of design decisions on performance.
- Key Message: Empower Yourself to Diagnose and Resolve Issues:
  - The session will equip you with the skills and knowledge to proactively tackle performance problems.

Connection Details

Server:

dataconwestus2.database.windows.net

User & Password (SQL Login)

Assigned in the post-it that you have.

Database:

PerfTroubleshootingDB

SLO:

Serverless, Gen5, 4 vCores

Introduction to Query Store & Copilot

- **What** is Query Store and **why it matters**
- **Key differences:** On-prem vs Azure SQL
- Introduction to Azure SQL Copilot and SSMS Copilot
- ***Demo***: Ask Copilot to analyze query performance from yesterday

Query Store Internals & Diagnostics

- Understanding **query_id, plan_id, runtime stats, wait stats**
- Detecting **regressions**, **most expensive** queries, and **anomalies**
- Forcing plans: **when and why**
- Query Store across **multiple databases**
- *Demo:* **Investigate a degraded query using SSMS or Azure Copilot**

Hands-on Scenarios with Python Simulation

- **High CPU** (single and multi-thread)
- **CXPACKET** (query parallel execution)
- **Different Execution Plan** (parameter sniffing)
- **Command Timeout** with Retry Logic
  - **ODBC Connection Reuse after Error**
- **Simulation High Data Read and Write.**
- **High Concurrency**
- **Additional Topic - Coding Stuff.**

Python Code

Session Context:

Introduction to a Python application that simulates common database bad performance scenarios.

Focus on identifying and resolving these issues collaboratively.

Tools We'll Use:

Python: The programming language used to develop the application and simulate scenarios.

Query Data Store (QDS): A tool for monitoring and analyzing query performance in SQL Server.

SQL Copilot: An AI-powered assistant that helps diagnose and solve database issues.

Importance of the Topic:

Performance issues can significantly impact applications and user experience.

Collaboration between developers and DBAs is crucial for efficiently identifying and resolving these problems.

Expectations for Attendees:

No prior experience with the mentioned tools is required.

The session will be practical and focused on real-world examples.

Participation and questions are encouraged throughout the presentation.

Python Connection Details

Authentication:

ODBC Driver 18

username/password (via credentials.txt)

Connection Handling:

Defined in ConnectToTheDB() in Python code.

Supports retry, timeout, and logging

Tables involved

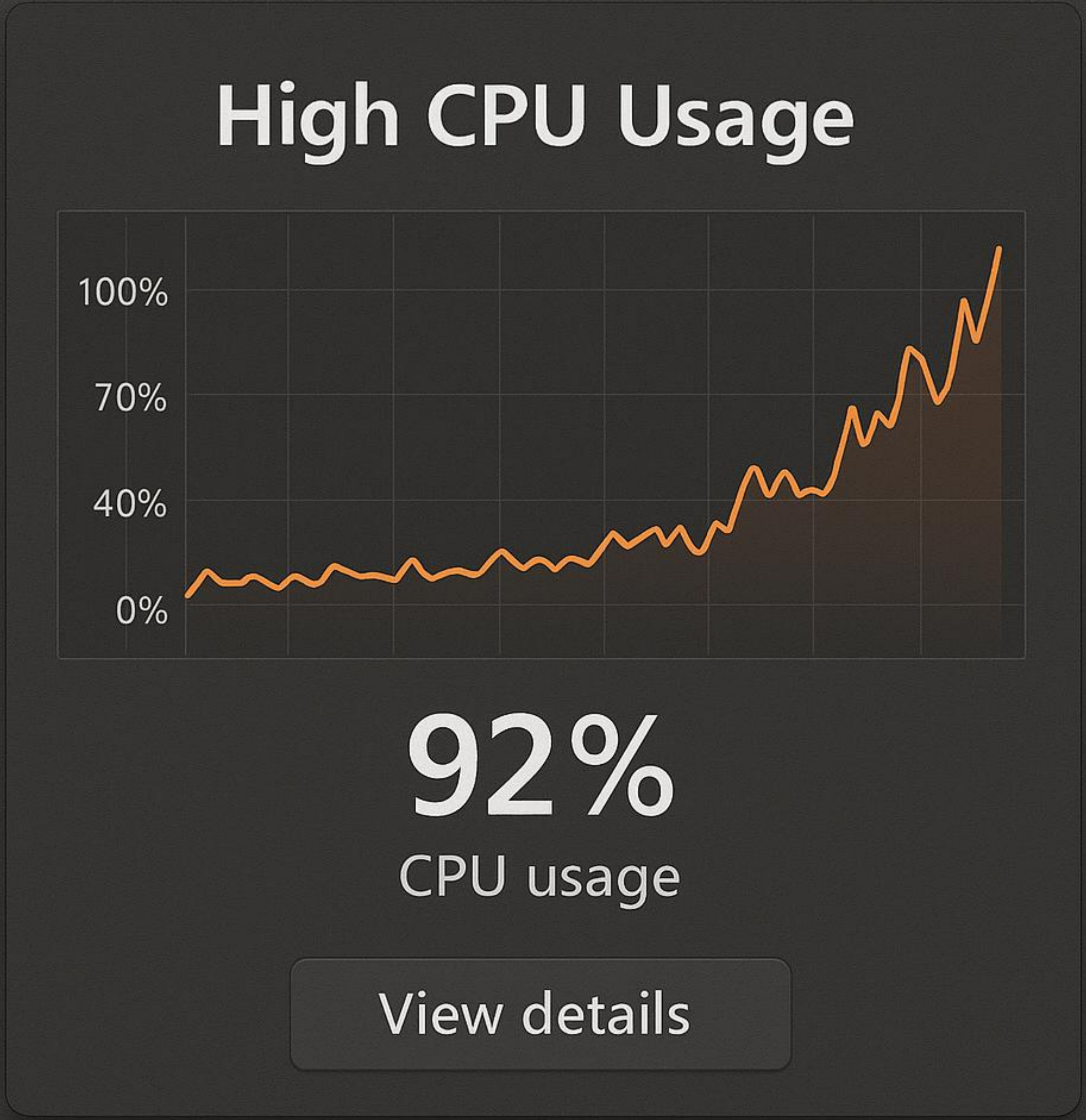
Schema	Table	Description & Usage
Dbo	**Inventory**	Used for **high concurrency simulation** with locking
Dbo	**MS_TableA_MS / MS_TableB_MS**	Used in **deadlock simulation** (simulate_deadlock)
Dbo	**Notes**	Accessed via **GiveNotes** for **plan regression testing**
Dbo	**Products2**	Queried in **chatty app simulation** with optional caching
MSxyzTest	**_x_y_z_MS_HighAsyncNetworkIO**	Used to **simulate network latency** issues
MSxyzTest	**_x_y_z_MS_HighBulkInsert**	Prepared for **bulk insert scenarios**
MSxyzTest	**_x_y_z_MS_HighCPU**	Main target for **High CPU test** (looping query with TextToSearch)
MSxyzTest	**_x_y_z_MS_HighCXPacket**	Used for **CXPACKET** scenarios with complex sort/filter logic
MSxyzTest	**_x_y_z_MS_HighDATAIO / HighDATAIOBlocks**	Referenced in **inefficient data access patterns**
MSxyzTest	**_x_y_z_MS_HighLocks**	Included for potential future use in **lock contention scenarios**
MSxyzTest	**_x_y_z_MS_HighLogIO**	Placeholder for **log IO stress scenarios**
MSxyzTest	**_x_y_z_MS_HighTempDB**	Used in **tempdb** contention simulations

Python Code Modules Ovreview (Database Workload)

- **RunHighCPU()**
  - Executes **many CPU-intensive queries using VARCHAR or WCHAR**
- **RunDifferentExecutionPlan()**
  - **Calls stored procedure with random inputs to vary plan**
- **RunCommandTimeout()**
  - **Demonstrates timeout + retry logic pattern**
- **RunHighNetworkIO**
  - **High Network Latency and DataIO Write (PowerShell)**
- **RunConcurrency()**
  - **A lot of blocking issues.**

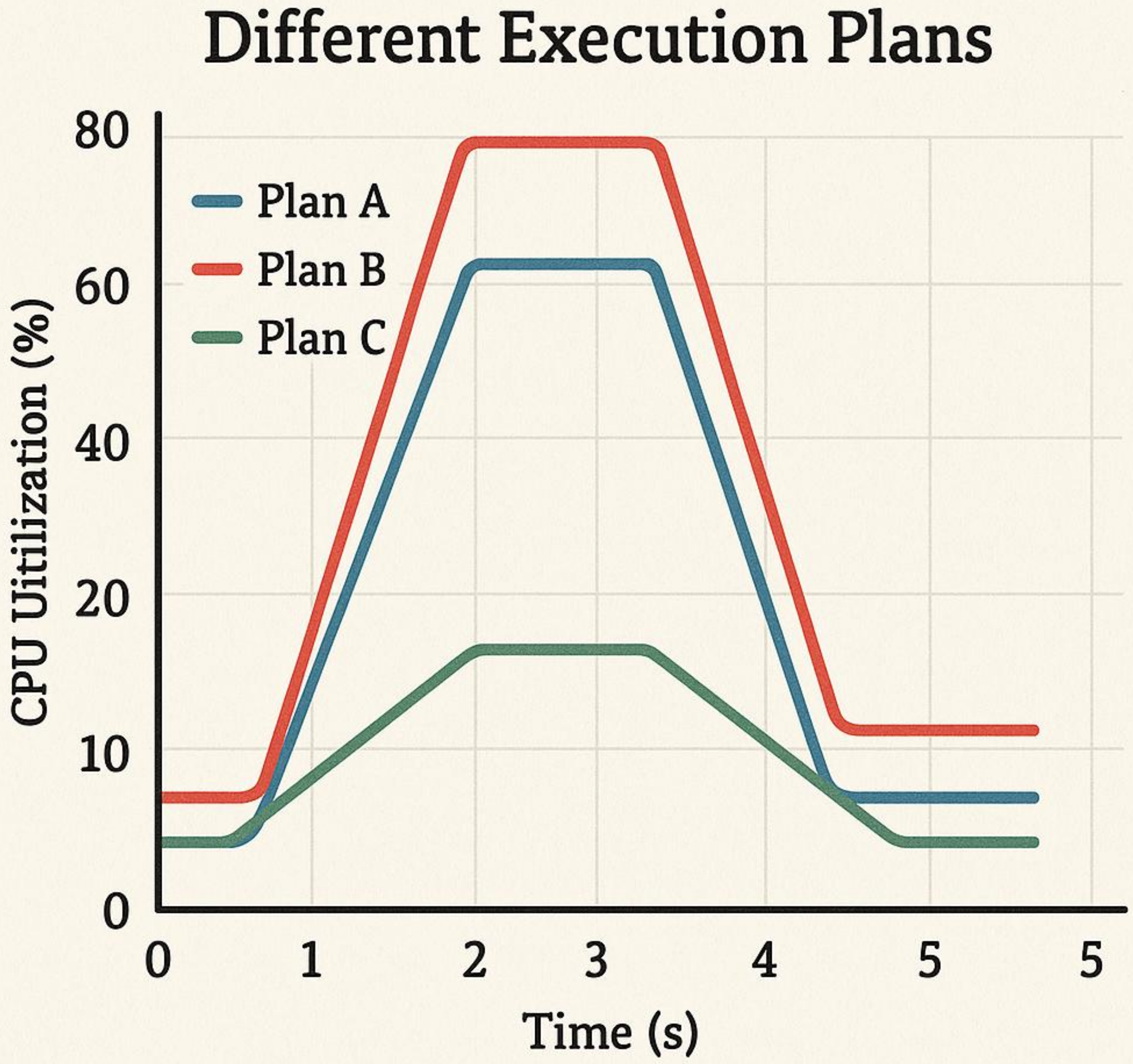
Hands-On-Lab I

High CPU and CXPacket



Hands-On-Lab II

Different Execution Plans



Hands-On-Lab III

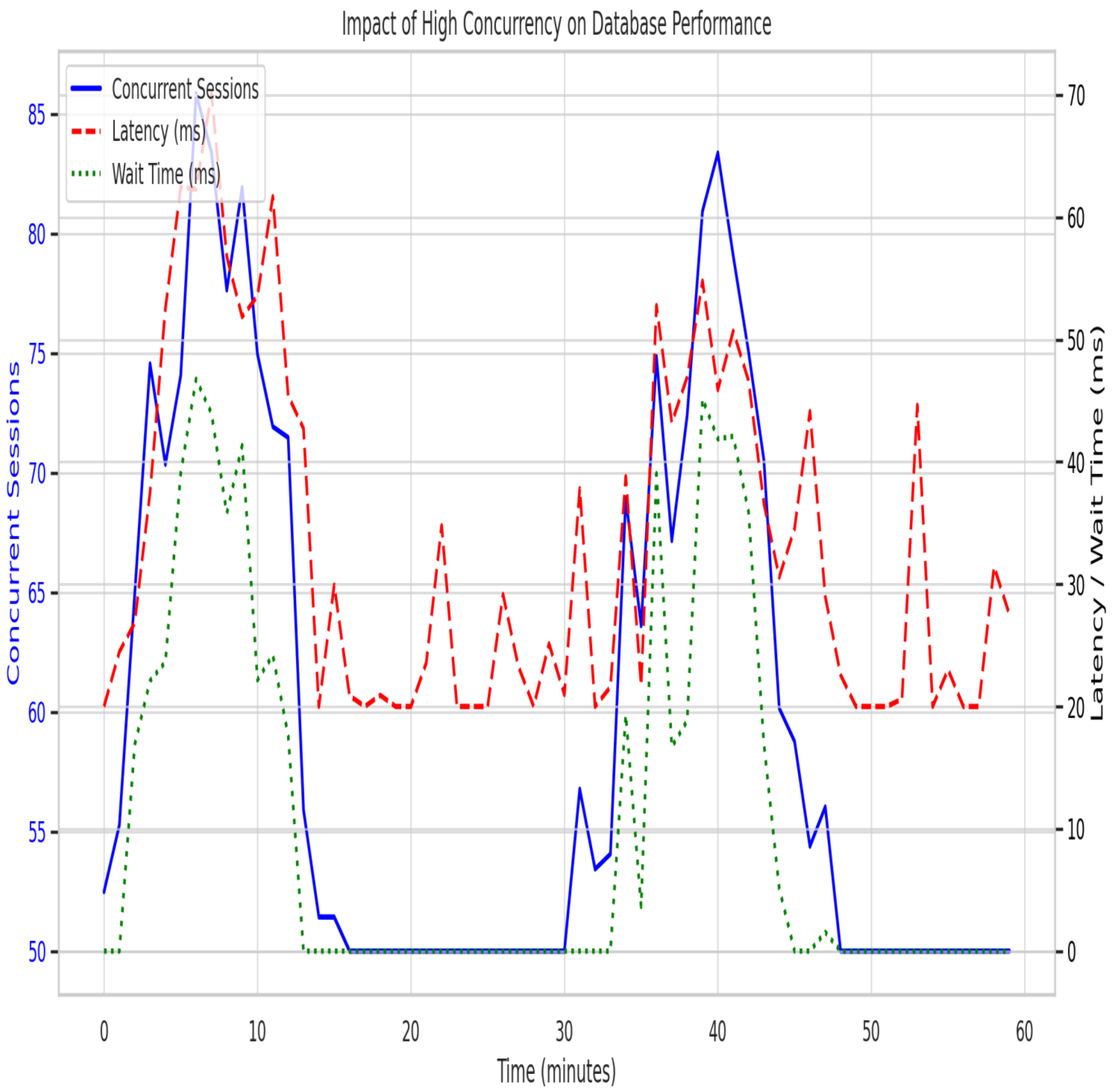
Command Timeout vs
Connection Timeout

Hands-On-Lab IV

PowerShell Script bulkInsert and DataIO

Hands-On-Lab V

Concurrency



Hands-On-Lab VI

Finding Performance Issue (Portal, QDS and Copilot).

Hands-On-Lab VII

QDS Summary Across Elastic database Pool

Support Insights & Best Practices

- **Real-world support** case learnings
- Query Store for **proactive health monitoring**
- When to use **Copilot vs traditional tools**

Python Code Modules Ovreview (Coding Best Practices)

- **ConnectToTheDB()**
  - **Central function for all DB connections**, manages retries and logs
- **run_connection_benchmark()**
  - Stress test for **connection time and limits**
- **RunSimpleInefficiencyWithTiming()**
  - **Efficient vs inefficient data processing example**
- **create_guiNumberOfExecutions()**
  - Stress test for execution number of a limited of threads.

Q&A + Wrap Up

- **Final** questions
- **Sharing resources and GitHub repo**
- What will you apply after this session?

Microsoft

Thank you.

Jose Manuel Jurado Diaz
Escalation Engineer
Microsoft

Juan Moreno Romo
Support Escalation Engineer
Microsoft

Queries right now? *Shoot them!*
Queries in future? *Ping them!*

```
-- How can I obtain the query with ID 1?
-- Which queries are consuming the most CPU in the database in the last hour?
-- Which queryID Text are consuming the most CPU in the database in the last hour, show me the query text?
-- How can I optimize the query with ID 4?
-- How can I identify and add missing indexes for my database?
-- Are there any missing index suggestions for the high CPU-consuming queries?
-- Which queries are being canceled due to timeouts?
-- Which queries are exhibiting common performance antipatterns?
-- What is the average and maximum duration of the queries canceled by timeouts?
-- Which queries are using inefficient joins or subqueries?
-- Can you provide a list of missing indexes in my database?
-- What are the most common wait times in my database?
-- Which queries are using forced execution plans?
-- Is there any command timeout in the database?
-- Is there any query with different execution plans?
-- What are the reasons for multiple execution plans being generated for the same query?
-- Is parameter sniffing causing performance issues with queries that have multiple execution plans?
-- give the TSQL command statement to identify the query_id where the sql statement is "SELECT count(Name),name FROM Notes where ID<@n group by Name"
-- give the TSQL command statement to identify the query_id where the sql statement is like "SELECT count(Name),name FROM Notes where ID<@n group by Name"
-- How many times has the query with id 4 been executed?
-- Show me the execution history of query ID 4.
-- Can you show the execution plan for query ID 4?
-- Are there any queries using unnecessary SELECT * operations?
-- Can you provide a detailed list of wait times and their deltas?
-- Can you provide the text of the queries that are consuming the most CPU?
```