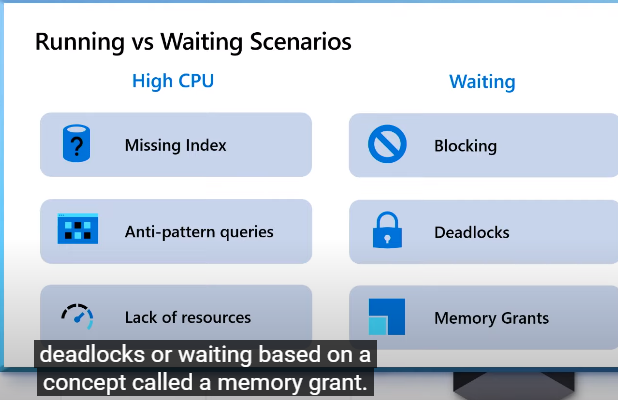
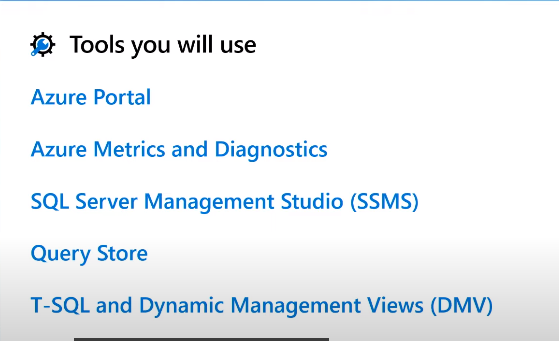
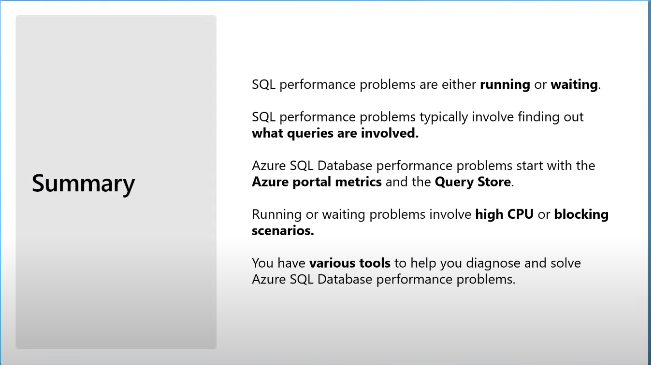
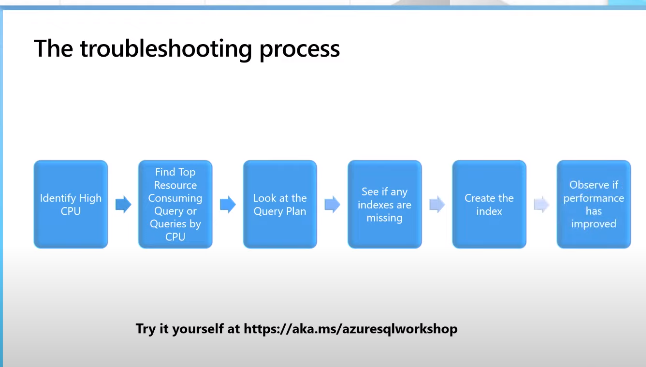
# **Introduction to Performance Troubleshooting [1 of 7] | Azure SQL for beginners**

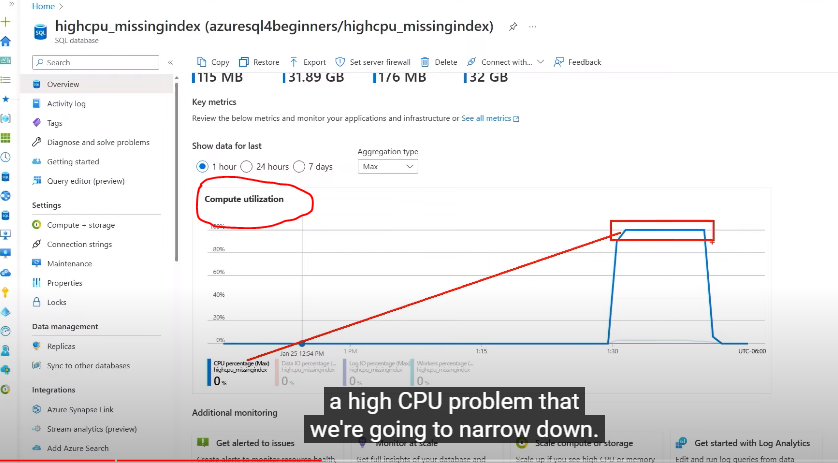






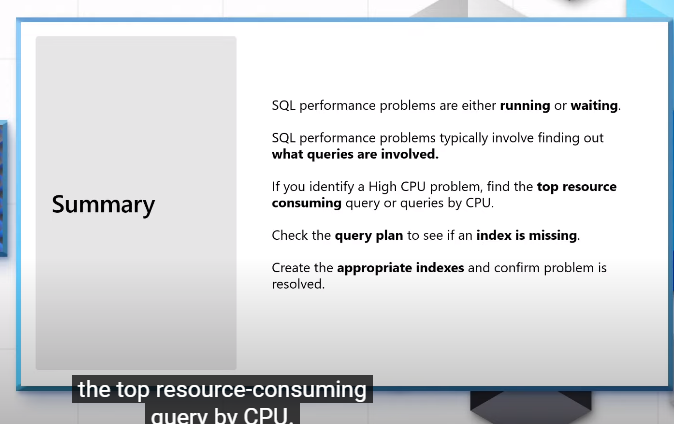
# **High CPU due to a Missing Index [2 of 7] | Azure SQL for beginners**



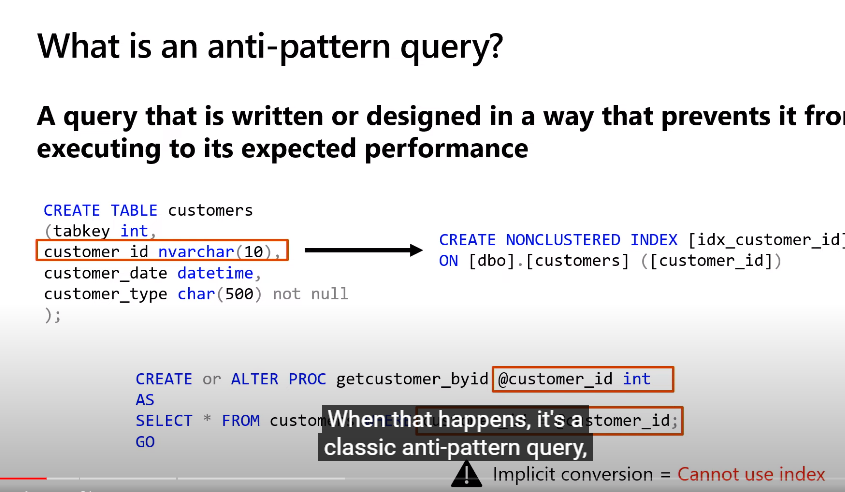


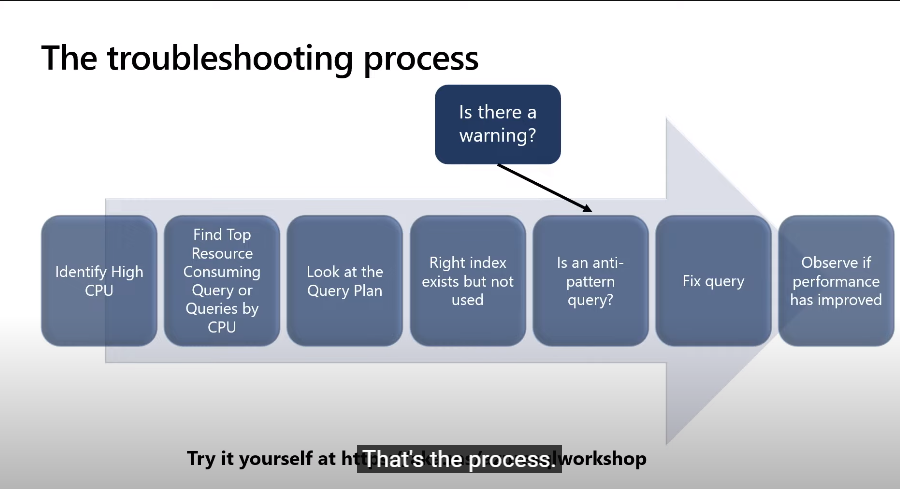
After creating a missing index as per the Sql server recommendation (took recommendation from Query store), below is the improvement

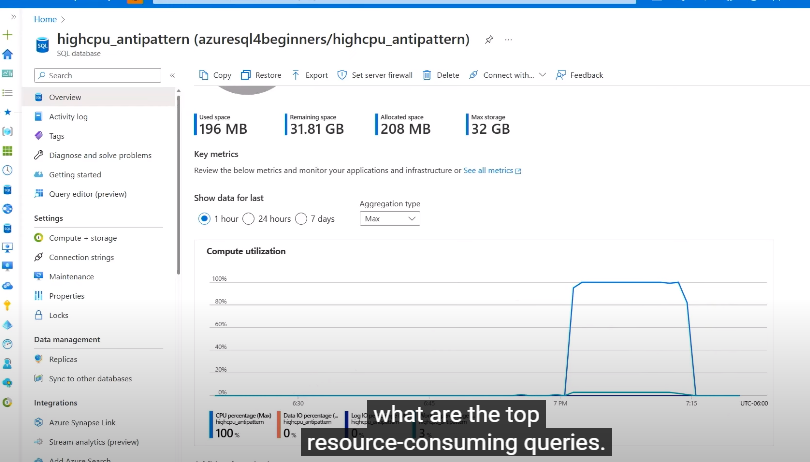


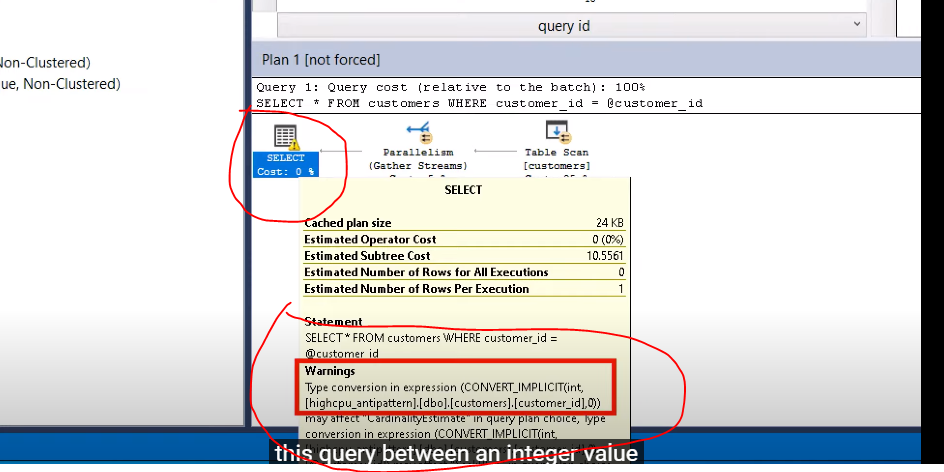


# **High CPU due to anti-pattern queries [3 of 7] | Azure SQL for beginners**

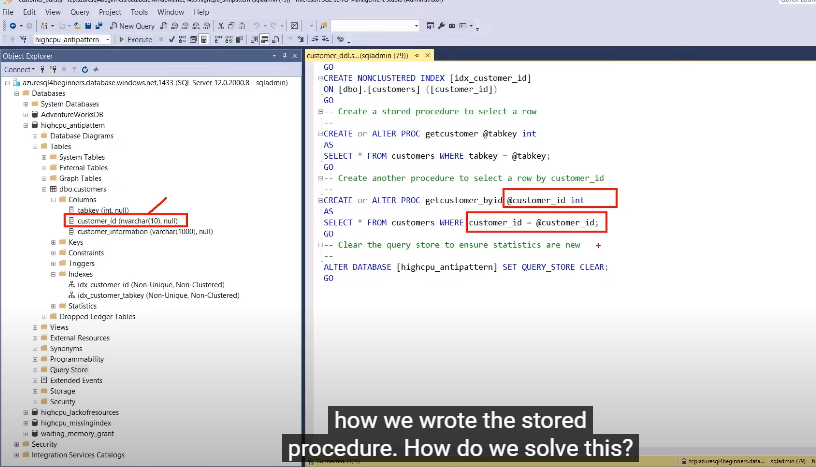






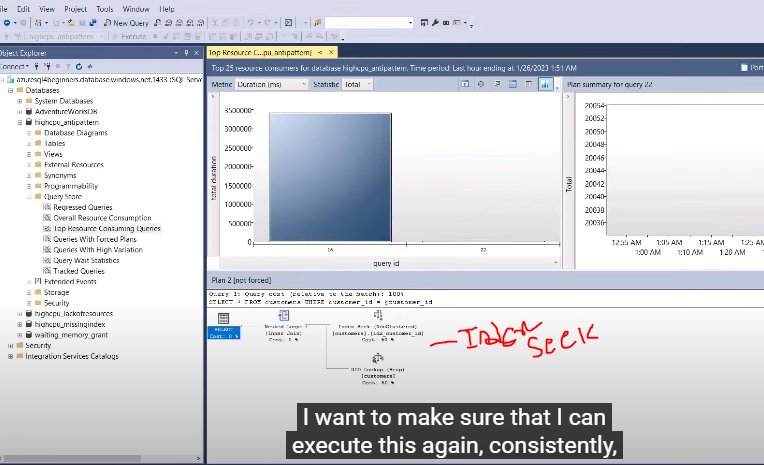


Carefully look at warning symbol in the execution plan

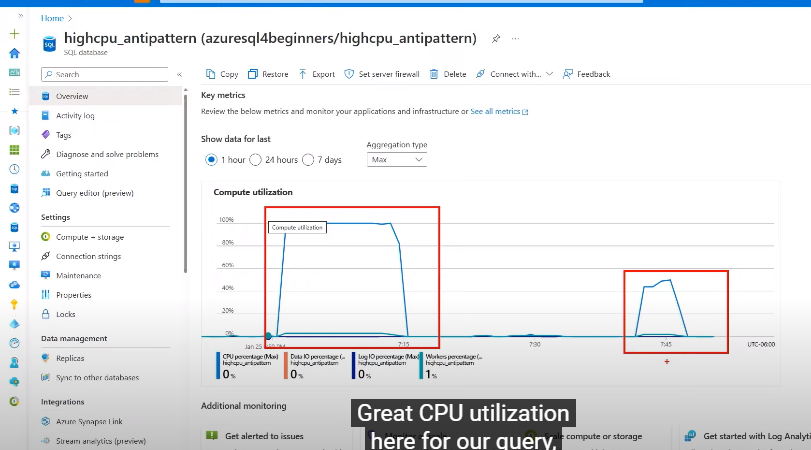


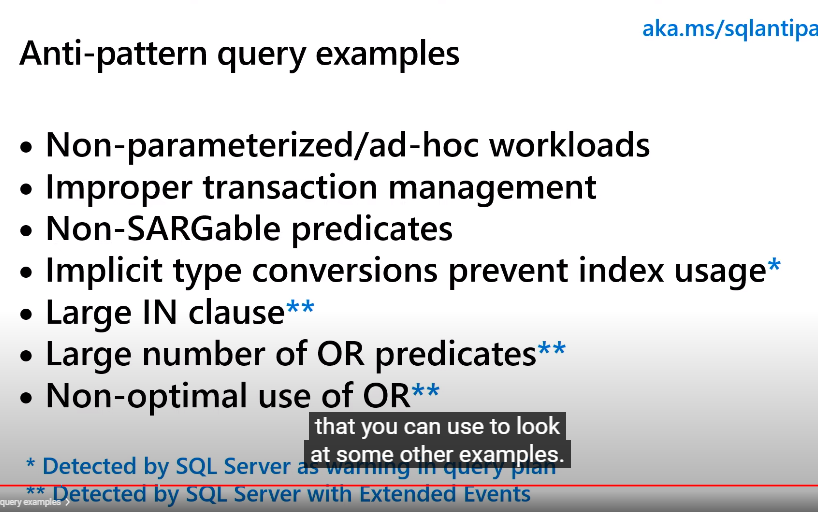
In the above snip, we can see on the left-hand side, Customer id column is created in nvarchar, but the SP is created using INT .. This will cause anti pattern query …

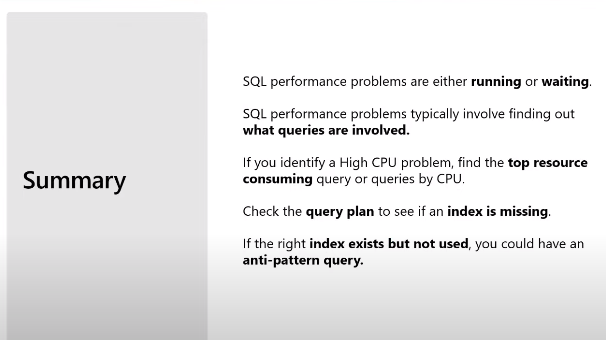
Once you have created the SP using correct column type, your execution plan will change as shown below (This is how performance is improving)



After implementing the fix, see how CPU utilization is improved





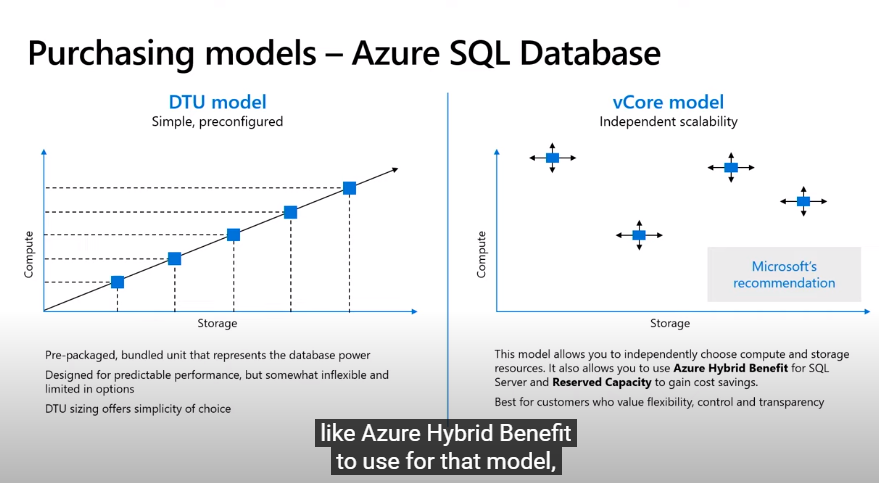


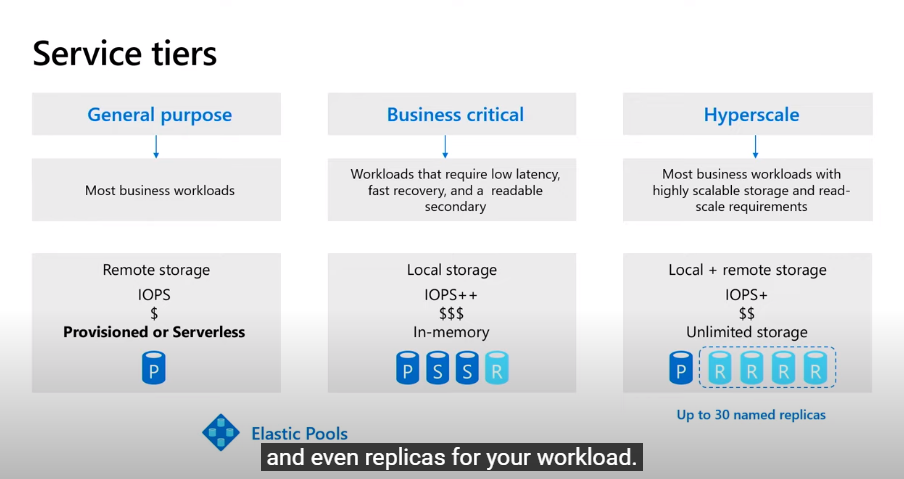
**Indexes** are created on tables or views. Indexes can be likened to a book directory and improve the speed of searches. Some examples of different types of indexes are clustered indexes, non-cluster indexes and column store indexes.

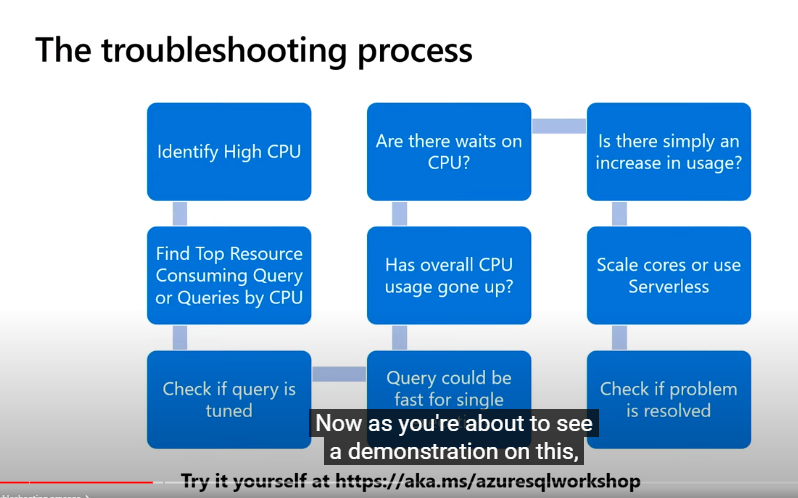
An **index scan** is when SQL Server **reads all the data in the index pages**. The cost of an index scan is very expensive for the SQL Server Engine.

An **index seek** is when SQL Server **reads only matching data in the index pages**. This method is more efficient for query performance because it will reduce IO and time consumption.

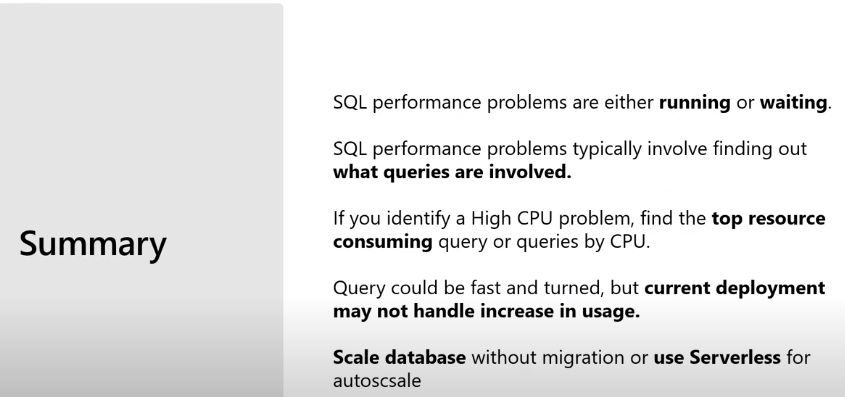
# **High CPU due to lack of resources [4 of 7] | Azure SQL for beginners**



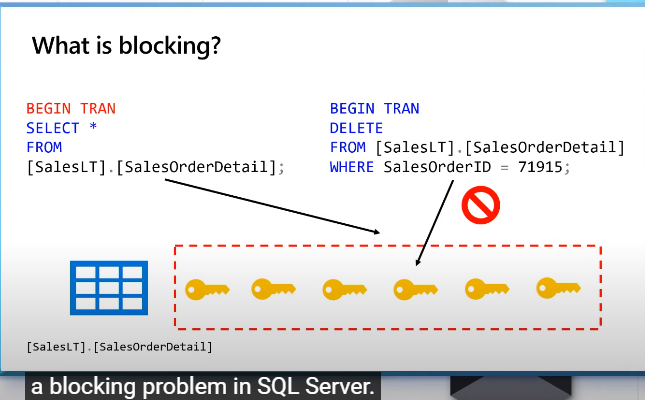


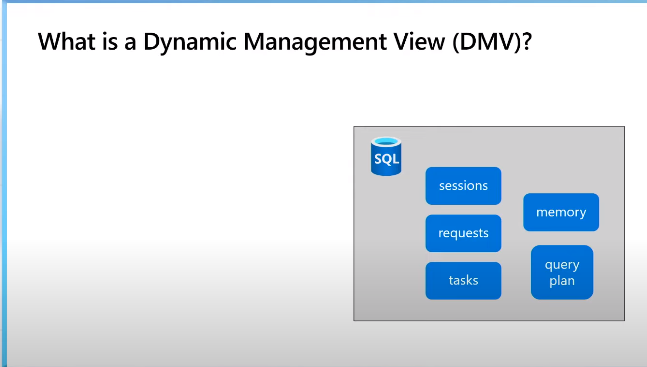


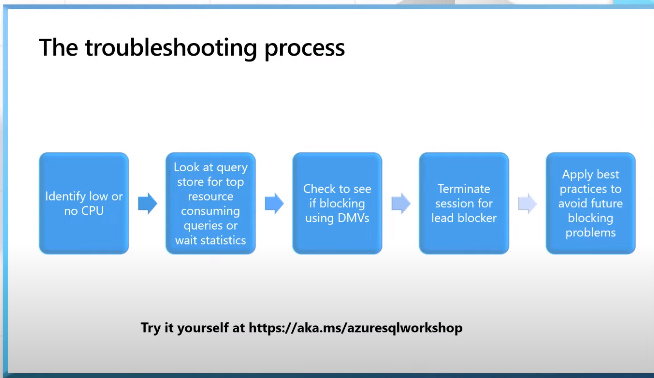
Adding more cores will fix the issue, if you are not sure how many cores to increase ..serverless model help us to solve this High CPU problem due to lack of resources



# **Waiting due to Blocking [5 of 7] | Azure SQL for beginners**







Steps in troubleshooting:

1. Identify the main blocking session (head blocker)
2. Find the query and transaction that is causing the blocking (what is holding locks for a prolonged period)
3. Analyze/understand why the prolonged blocking occurs
4. Resolve blocking issue by redesigning query and transaction

**Tsql to identify blocking: (https://learn.microsoft.com/en-in/azure/azure-sql/database/understand-resolve-blocking?view=azuresql-db)**

WITH cteHead ( session\_id,request\_id,wait\_type,wait\_resource,last\_wait\_type,is\_user\_process,request\_cpu\_time

,request\_logical\_reads,request\_reads,request\_writes,wait\_time,blocking\_session\_id,memory\_usage

,session\_cpu\_time,session\_reads,session\_writes,session\_logical\_reads

,percent\_complete,est\_completion\_time,request\_start\_time,request\_status,command

,plan\_handle,sql\_handle,statement\_start\_offset,statement\_end\_offset,most\_recent\_sql\_handle

,session\_status,group\_id,query\_hash,query\_plan\_hash)

AS ( SELECT sess.session\_id, req.request\_id, LEFT (ISNULL (req.wait\_type, ''), 50) AS 'wait\_type'

, LEFT (ISNULL (req.wait\_resource, ''), 40) AS 'wait\_resource', LEFT (req.last\_wait\_type, 50) AS 'last\_wait\_type'

, sess.is\_user\_process, req.cpu\_time AS 'request\_cpu\_time', req.logical\_reads AS 'request\_logical\_reads'

, req.reads AS 'request\_reads', req.writes AS 'request\_writes', req.wait\_time, req.blocking\_session\_id,sess.memory\_usage

, sess.cpu\_time AS 'session\_cpu\_time', sess.reads AS 'session\_reads', sess.writes AS 'session\_writes', sess.logical\_reads AS 'session\_logical\_reads'

, CONVERT (decimal(5,2), req.percent\_complete) AS 'percent\_complete', req.estimated\_completion\_time AS 'est\_completion\_time'

, req.start\_time AS 'request\_start\_time', LEFT (req.status, 15) AS 'request\_status', req.command

, req.plan\_handle, req.[sql\_handle], req.statement\_start\_offset, req.statement\_end\_offset, conn.most\_recent\_sql\_handle

, LEFT (sess.status, 15) AS 'session\_status', sess.group\_id, req.query\_hash, req.query\_plan\_hash

FROM sys.dm\_exec\_sessions AS sess

LEFT OUTER JOIN sys.dm\_exec\_requests AS req ON sess.session\_id = req.session\_id

LEFT OUTER JOIN sys.dm\_exec\_connections AS conn on conn.session\_id = sess.session\_id

)

, cteBlockingHierarchy (head\_blocker\_session\_id, session\_id, blocking\_session\_id, wait\_type, wait\_duration\_ms,

wait\_resource, statement\_start\_offset, statement\_end\_offset, plan\_handle, sql\_handle, most\_recent\_sql\_handle, [Level])

AS ( SELECT head.session\_id AS head\_blocker\_session\_id, head.session\_id AS session\_id, head.blocking\_session\_id

, head.wait\_type, head.wait\_time, head.wait\_resource, head.statement\_start\_offset, head.statement\_end\_offset

, head.plan\_handle, head.sql\_handle, head.most\_recent\_sql\_handle, 0 AS [Level]

FROM cteHead AS head

WHERE (head.blocking\_session\_id IS NULL OR head.blocking\_session\_id = 0)

AND head.session\_id IN (SELECT DISTINCT blocking\_session\_id FROM cteHead WHERE blocking\_session\_id != 0)

UNION ALL

SELECT h.head\_blocker\_session\_id, blocked.session\_id, blocked.blocking\_session\_id, blocked.wait\_type,

blocked.wait\_time, blocked.wait\_resource, h.statement\_start\_offset, h.statement\_end\_offset,

h.plan\_handle, h.sql\_handle, h.most\_recent\_sql\_handle, [Level] + 1

FROM cteHead AS blocked

INNER JOIN cteBlockingHierarchy AS h ON h.session\_id = blocked.blocking\_session\_id and h.session\_id!=blocked.session\_id --avoid infinite recursion for latch type of blocking

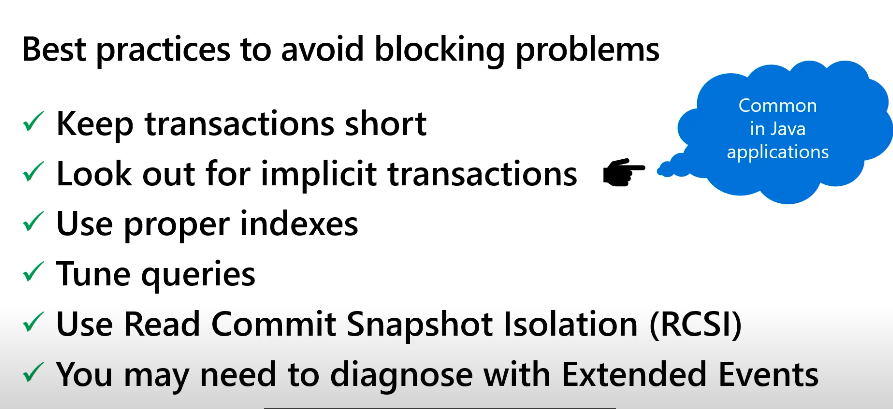
WHERE h.wait\_type COLLATE Latin1\_General\_BIN NOT IN ('EXCHANGE', 'CXPACKET') or h.wait\_type is null

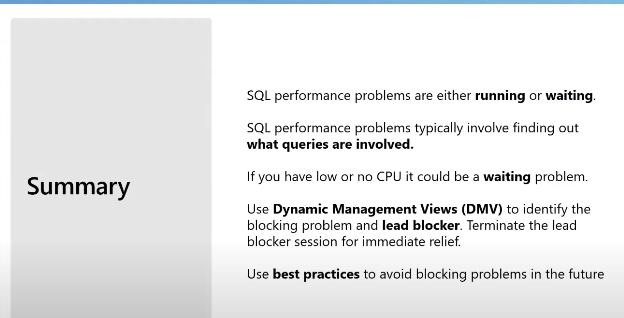
)

SELECT bh.\*, txt.text AS blocker\_query\_or\_most\_recent\_query

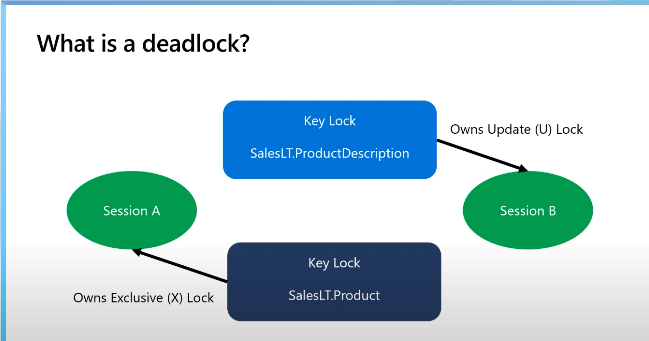
FROM cteBlockingHierarchy AS bh

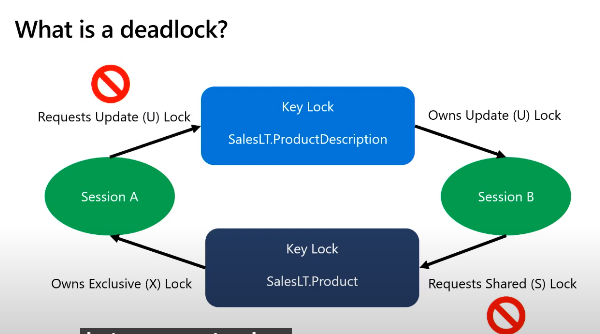
OUTER APPLY sys.dm\_exec\_sql\_text (ISNULL ([sql\_handle], most\_recent\_sql\_handle)) AS txt;

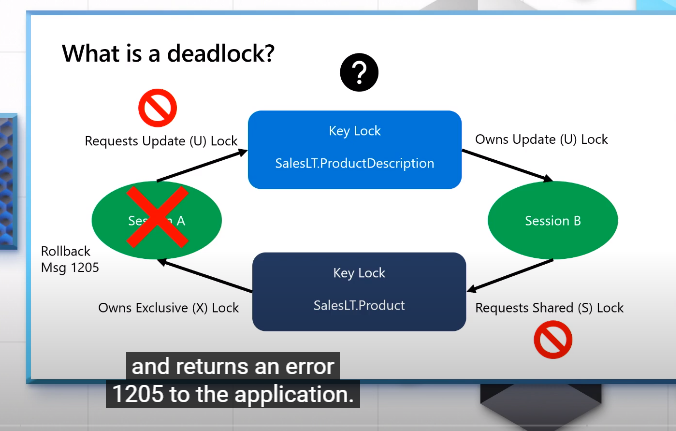


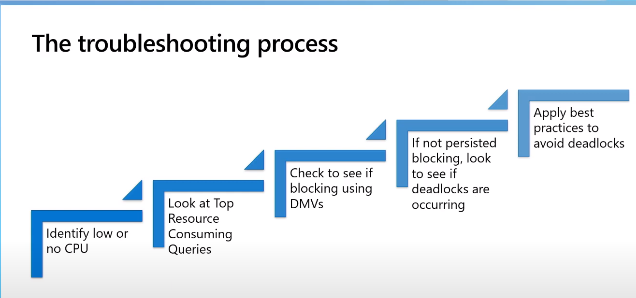


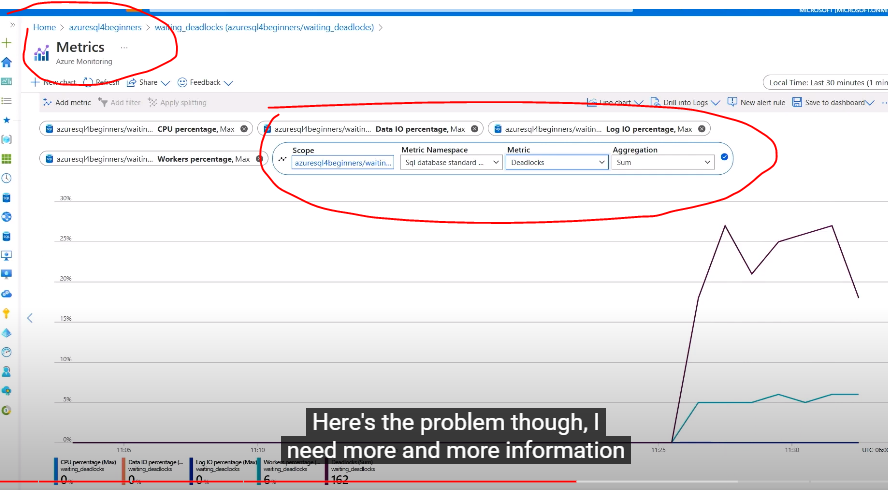
# **Waiting due to deadlocks [6 of 7] | Azure SQL for beginners**

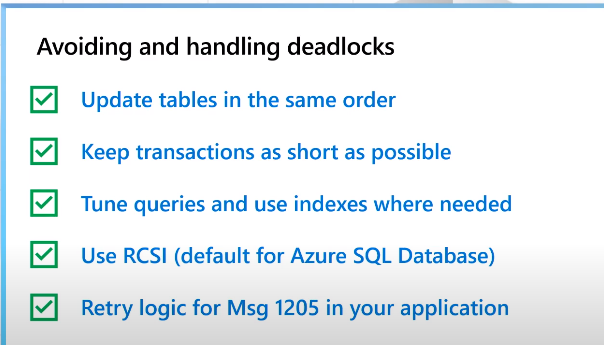


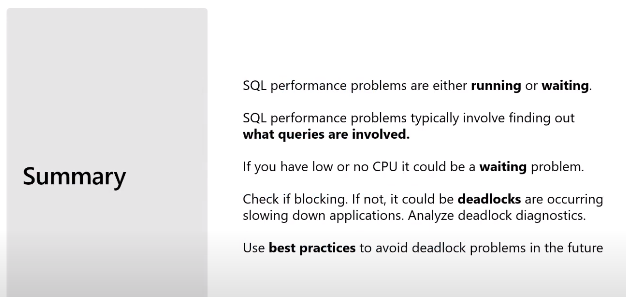




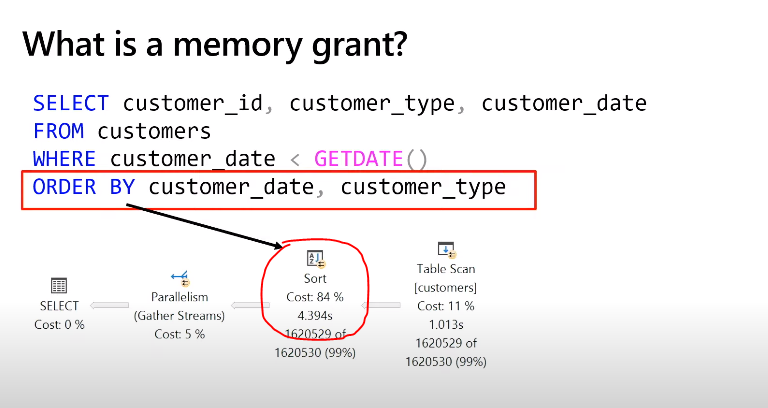


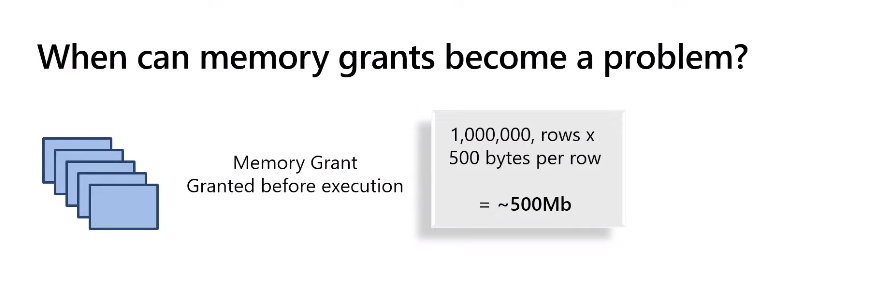


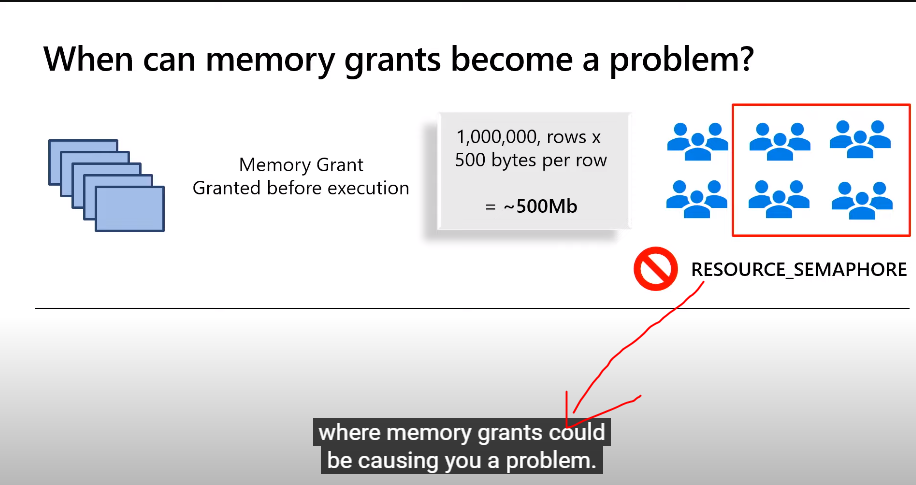


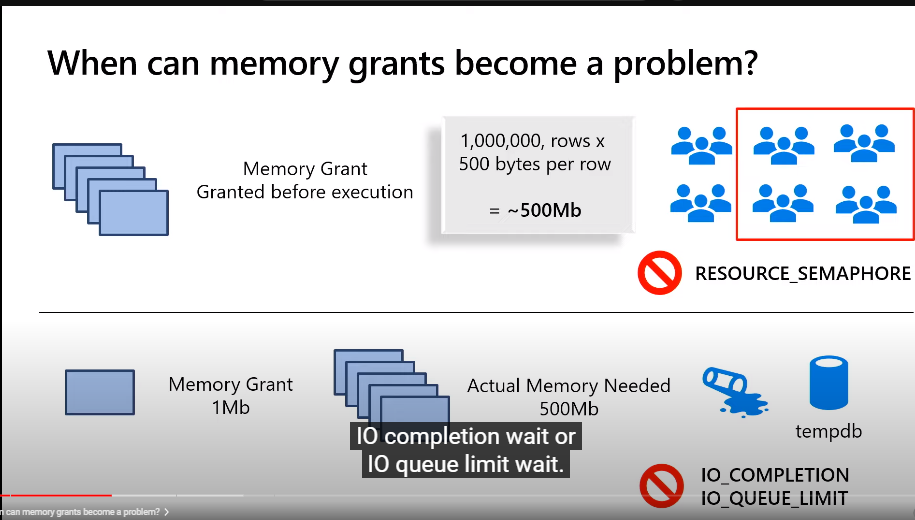


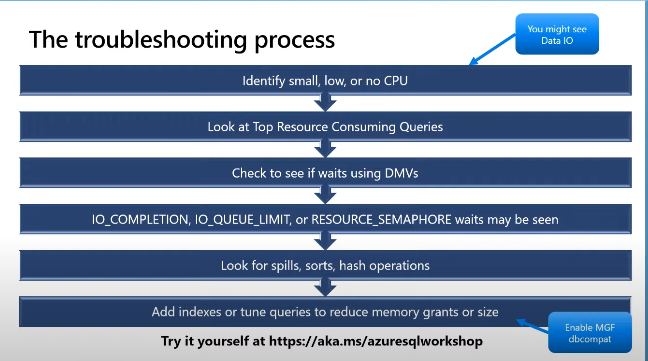
# **Waiting due to memory grants [7 of 7] | Azure SQL for beginners**

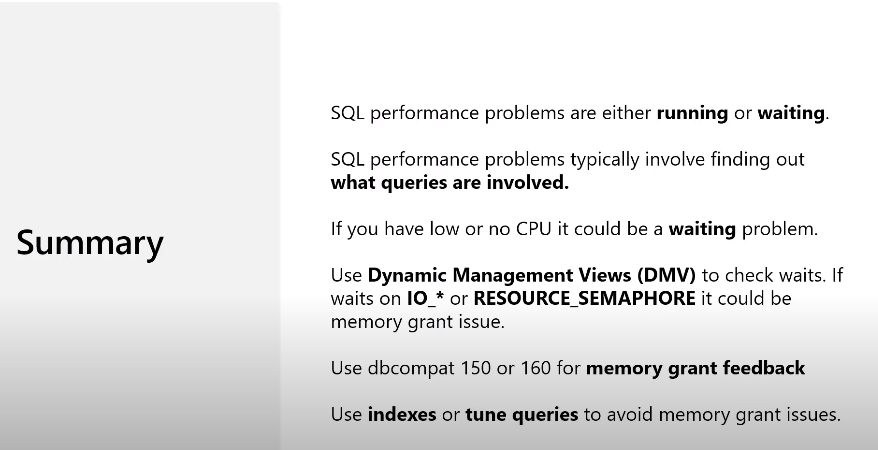












Tsql to check memory related :

SELECT CONVERT (varchar(30), GETDATE(), 121) as runtime

, r.session\_id

, r.wait\_time

, r.wait\_type

, mg.request\_time

, mg.grant\_time

, mg.requested\_memory\_kb

/ 1024 requested\_memory\_mb

, mg.granted\_memory\_kb

/ 1024 AS granted\_memory\_mb

, mg.required\_memory\_kb

/ 1024 AS required\_memory\_mb

, max\_used\_memory\_kb

/ 1024 AS max\_used\_memory\_mb

, rs.pool\_id as resource\_pool\_id

, mg.query\_cost

, mg.timeout\_sec

, mg.resource\_semaphore\_id

, mg.wait\_time\_ms AS memory\_grant\_wait\_time\_ms

, CASE mg.is\_next\_candidate

WHEN 1 THEN 'Yes'

WHEN 0 THEN 'No'

ELSE 'Memory has been granted'

END AS 'Next Candidate for Memory Grant'

, r.command

, ltrim(rtrim(replace(replace (substring (q.text, 1, 1000), char(10), ' '), char(13), ' '))) [text]

, rs.target\_memory\_kb

/ 1024 AS server\_target\_grant\_memory\_mb

, rs.max\_target\_memory\_kb

/ 1024 AS server\_max\_target\_grant\_memory\_mb

, rs.total\_memory\_kb

/ 1024 AS server\_total\_resource\_semaphore\_memory\_mb

, rs.available\_memory\_kb

/ 1024 AS server\_available\_memory\_for\_grants\_mb

, rs.granted\_memory\_kb

/ 1024 AS server\_total\_granted\_memory\_mb

, rs.used\_memory\_kb

/ 1024 AS server\_used\_granted\_memory\_mb

, rs.grantee\_count AS successful\_grantee\_count

, rs.waiter\_count AS grant\_waiters\_count

, rs.timeout\_error\_count

, rs.forced\_grant\_count

, mg.dop

, r.blocking\_session\_id

, r.cpu\_time

, r.total\_elapsed\_time

, r.reads

, r.writes

, r.logical\_reads

, r.row\_count

, s.login\_time

, d.name

, s.login\_name

, s.host\_name

, s.nt\_domain

, s.nt\_user\_name

, s.status

, c.client\_net\_address

, s.program\_name

, s.client\_interface\_name

, s.last\_request\_start\_time

, s.last\_request\_end\_time

, c.connect\_time

, c.last\_read

, c.last\_write

, qp.query\_plan

FROM sys.dm\_exec\_requests r

INNER JOIN sys.dm\_exec\_connections c

ON r.connection\_id = c.connection\_id

INNER JOIN sys.dm\_exec\_sessions s

ON c.session\_id = s.session\_id

INNER JOIN sys.databases d

ON r.database\_id = d.database\_id

INNER JOIN sys.dm\_exec\_query\_memory\_grants mg

ON s.session\_id = mg.session\_id

INNER JOIN sys.dm\_exec\_query\_resource\_semaphores rs

ON mg.resource\_semaphore\_id = rs.resource\_semaphore\_id

CROSS APPLY sys.dm\_exec\_sql\_text (r.sql\_handle ) AS q

CROSS APPLY sys.dm\_exec\_query\_plan(mg.plan\_handle) qp

OPTION (MAXDOP 1, LOOP JOIN )