

Azure Synapse Assessment Query Execution Logs Extraction and Source Code Prerequisites

Contents

1. Introduction	3
2. Assessment Process	3
2.1 Databases	3
2.2 Database Volume	3
2.3 High Data Volume Tables	4
2.4 Users	7
2.5 User Tables	8
2.6 Objects	8
2.7 Server Configuration and Concurrent Number of Queries Allowed.	8
2.8 Average, Max CPU, IO and Storage Stats	9
2.9 Total I/O Usage by Database	10
2.10 Top CPU and I/O Consuming Queries	10
2.11 Top CPU and I/O Consuming Procedures	13
2.12 Data for Partitioning and Bucketing	14
3. Source Code Assessment	18

1. Introduction

LeapLogic's Assessment profiles existing inventory identifies workload complexity, interdependencies, and provides comprehensive actionable recommendations for migration to modern data platform.

Assessment checklist

Information	Format	Availability?
Database Object Count	File Export in text format	Choose an item.
Database Volume	File Export in text format	Choose an item.
High Data Volume Tables	File Export in text format	Choose an item.
Total Number of Users	File Export in text format	Choose an item.
Total Number of User Tables	File Export in text format	Choose an item.
Objects	File Export in text format	Choose an item.
Total I/O Usage by Database	File Export in text format	Choose an item.
Top CPU and I/O Consuming Queries	File Export in text format	Choose an item.
Top CPU and I/O Consuming Procedures	File Export in text format	Choose an item.
Top CPU and I/O Consuming Functions	File Export in text format	Choose an item.
Data for Partitioning and Bucketing	File Export in text format	Choose an item.
Source Code Assessment	File Export in text format	Choose an item.

Follow the steps in section 2, 3 to collect the required information for assessment.

2. Assessment Process

All the environment specific variables are highlighted in the document in **Green**. The start and end date should be consistent for all the queries.

2.1 Databases

This query provides the total number of databases. Execute it in a master database and save the results in CSV format.

```
--To be run under master database
select count(*) from sys.databases
```

2.2 Database Volume

This query provides the database volume in MBs. Execute it in a master database and save the results in CSV format.

```

-- To be run under master database

DECLARE @Database TABLE
(
    database_nm VARCHAR(50),
    log_write DECIMAL(8,2),
    row_size_mb DECIMAL(8,2),
    total_size_mb DECIMAL(8,2)
)
INSERT INTO @Database
SELECT
    database_name = CAST(database_name AS VARCHAR(50))
    ,log_write_secs = max(avg_log_write_percent)
    , row_size_mb = max(storage_in_megabytes)
    ,total_size_mb = max(allocated_storage_in_megabytes)

FROM sys.resource_stats rs inner join sys.databases db on rs.database_name = db.name
where sku='DW'
GROUP BY database_name
select * FROM @Database

```

2.3 High Data Volume Tables

This query provides the result for tables with a high data volume. Execute it for all the databases that need to be migrated and save the results in CSV format.

```

--- to run under all the DB of Synapse
WITH base
AS
(
    SELECT
        GETDATE() AS [execution_time]
        , DB_NAME() AS [database_name]
        , s.name AS [schema_name]
        , t.name AS [table_name]
        , QUOTENAME(s.name)+'.'+QUOTENAME(t.name) AS [two_part_name]
        , nt.[name] AS [node_table_name]
        , ROW_NUMBER() OVER(PARTITION BY nt.[name] ORDER BY (SELECT NULL)) AS
        [node_table_name_seq]
        , tp.[distribution_policy_desc] AS [distribution_policy_name]
        , c.[name] AS [distribution_column]
        , nt.[distribution_id] AS [distribution_id]

```

```

, i.[type] AS [index_type]
, i.[type_desc] AS [index_type_desc]
, nt.[pdw_node_id] AS [pdw_node_id]
, pn.[type] AS [pdw_node_type]
, pn.[name] AS [pdw_node_name]
, di.name AS [dist_name]
, di.position AS [dist_position]
, nps.[partition_number] AS [partition_nmbr]
, nps.[reserved_page_count] AS [reserved_space_page_count]
, nps.[reserved_page_count] - nps.[used_page_count] AS [unused_space_page_count]
, nps.[in_row_data_page_count]
+ nps.[row_overflow_used_page_count]
+ nps.[lob_used_page_count] AS [data_space_page_count]
, nps.[reserved_page_count]
- (nps.[reserved_page_count] - nps.[used_page_count])
- ([in_row_data_page_count]
+ [row_overflow_used_page_count]+[lob_used_page_count]) AS [index_space_page_count]
, nps.[row_count] AS [row_count]
from
sys.schemas s
INNER JOIN sys.tables t
ON s.[schema_id] = t.[schema_id]
INNER JOIN sys.indexes i
ON t.[object_id] = i.[object_id]
AND i.[index_id] <= 1
INNER JOIN sys.pdw_table_distribution_properties tp
ON t.[object_id] = tp.[object_id]
INNER JOIN sys.pdw_table_mappings tm
ON t.[object_id] = tm.[object_id]
INNER JOIN sys.pdw_nodes_tables nt
ON tm.[physical_name] = nt.[name]
INNER JOIN sys.dm_pdw_nodes pn
ON nt.[pdw_node_id] = pn.[pdw_node_id]
INNER JOIN sys.pdw_distributions di
ON nt.[distribution_id] = di.[distribution_id]
INNER JOIN sys.dm_pdw_nodes_db_partition_stats nps
ON nt.[object_id] = nps.[object_id]
AND nt.[pdw_node_id] = nps.[pdw_node_id]
AND nt.[distribution_id] = nps.[distribution_id]
AND i.[index_id] = nps.[index_id]
LEFT OUTER JOIN (select * from sys.pdw_column_distribution_properties where distribution_ordinal = 1)
cdp

```

```

    ON t.[object_id] = cdp.[object_id]
LEFT OUTER JOIN sys.columns c
    ON cdp.[object_id] = c.[object_id]
    AND cdp.[column_id] = c.[column_id]
WHERE pn.[type] = 'COMPUTE'
)
, size
AS
(
SELECT
    [execution_time]
, [database_name]
, [schema_name]
, [table_name]
, [two_part_name]
, [node_table_name]
, [node_table_name_seq]
, [distribution_policy_name]
, [distribution_column]
, [distribution_id]
, [index_type]
, [index_type_desc]
, [pdw_node_id]
, [pdw_node_type]
, [pdw_node_name]
, [dist_name]
, [dist_position]
, [partition_nmbr]
, [reserved_space_page_count]
, [unused_space_page_count]
, [data_space_page_count]
, [index_space_page_count]
, [row_count]
, ([reserved_space_page_count] * 8.0) AS [reserved_space_KB]
, ([reserved_space_page_count] * 8.0)/1000 AS [reserved_space_MB]
, ([reserved_space_page_count] * 8.0)/1000000 AS [reserved_space_GB]
, ([reserved_space_page_count] * 8.0)/1000000000 AS [reserved_space_TB]
, ([unused_space_page_count] * 8.0) AS [unused_space_KB]
, ([unused_space_page_count] * 8.0)/1000 AS [unused_space_MB]
, ([unused_space_page_count] * 8.0)/1000000 AS [unused_space_GB]
, ([unused_space_page_count] * 8.0)/1000000000 AS [unused_space_TB]
, ([data_space_page_count] * 8.0) AS [data_space_KB]

```

```

, ([data_space_page_count] * 8.0)/1000 AS [data_space_MB]
, ([data_space_page_count] * 8.0)/1000000 AS [data_space_GB]
, ([data_space_page_count] * 8.0)/1000000000 AS [data_space_TB]
, ([index_space_page_count] * 8.0) AS [index_space_KB]
, ([index_space_page_count] * 8.0)/1000 AS [index_space_MB]
, ([index_space_page_count] * 8.0)/1000000 AS [index_space_GB]
, ([index_space_page_count] * 8.0)/1000000000 AS [index_space_TB]
FROM base
)
SELECT
    database_name
,   table_name
,   schema_name
,   SUM(row_count) as table_row_count
,   SUM(([reserved_space_page_count] * 8.0)/1000000) as table_reserved_space_GB
,   SUM(([data_space_page_count] * 8.0)/1000000) as table_data_space_GB
,   SUM(([unused_space_page_count] * 8.0)/1000000) as table_unused_space_GB
FROM
    base
GROUP BY
    database_name
,   schema_name
,   table_name
,   distribution_policy_name
,   distribution_column
,   index_type_desc
having SUM(([data_space_page_count] * 8.0)/1000000) >=10
ORDER BY
    table_reserved_space_GB desc

```

This SQL will collect databases with volume equal or above 10GB. **This step must be performed for all the databases.**

Note: The filter condition can be changed for collecting lower volumes.

2.4 Users

This query provides the total number of users. Execute it for all the databases that need to be migrated and save the results in CSV format.

```

---to be run under all the db of synapse
select
database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
type ,count(*) [NUM]
from sys.database_principals
group by type

```

2.5 User Tables

This query provides the total number of user tables. Execute it in a synapse database and save the results in CSV format.

```

--to be executed in all synapse db
select
database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
Name,type_desc,create_date,modify_date
from sys.tables

```

2.6 Objects

This query provides all the objects in a database. Execute it for all the databases that need to be migrated and save the results in CSV format.

```

--- To be executed in all synapse db

select
database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
Name,type_desc,create_date,modify_date
from sys.objects where is_ms_shipped=0

```

2.7 Server Configuration and Concurrent Number of Queries Allowed.

This query provides server configuration along with concurrent number of queries allowed. Execute it on master database and save the results in CSV format.

```

--To be run under master Database
create table #Concurrent_queries
(
    service_objectives VARCHAR(50),
    Max_concurrent_queries DECIMAL(8,2)
);

INSERT INTO #Concurrent_queries values ('DW100c',4);
INSERT INTO #Concurrent_queries values ('DW200c',8);

```



```

INSERT INTO #Concurrent_queries values ('DW300c',12);
INSERT INTO #Concurrent_queries values ('DW400c',16);
INSERT INTO #Concurrent_queries values ('DW500c',20);
INSERT INTO #Concurrent_queries values ('DW1000c',32);
INSERT INTO #Concurrent_queries values ('DW1500c',48);
INSERT INTO #Concurrent_queries values ('DW2000c',48);
INSERT INTO #Concurrent_queries values ('DW2500c',64);
INSERT INTO #Concurrent_queries values ('DW3000c',64);
INSERT INTO #Concurrent_queries values ('DW5000c',128);
INSERT INTO #Concurrent_queries values ('DW6000c',128);
INSERT INTO #Concurrent_queries values ('DW7500c',128);
INSERT INTO #Concurrent_queries values ('DW10000c',128);
INSERT INTO #Concurrent_queries values ('DW15000c',128);
INSERT INTO #Concurrent_queries values ('DW30000c',128);

SELECT db.name [Database]
,      ds.edition [Edition]
,      ds.service_objective [Service_Objective]
, cq.Max_concurrent_queries [Concurrent_queries]
FROM   sys.database_service_objectives AS ds
JOIN   sys.databases                AS db ON ds.database_id = db.database_id
JOIN   #Concurrent_queries AS cq on upper(cq.service_objectives) = upper(ds.service_objective)
;

```

2.8 Average, Max CPU, IO and Storage Stats

This query provides average, max CPU/IP and storage statistics. Execute it on master database and save the results in CSV format.

```

--To be run under master database
DECLARE @s datetime;
DECLARE @e datetime;
SET @s= DateAdd(d,-7,GetUTCDate());
SET @e= GETUTCDATE();

SELECT
    database_name,
    AVG(avg_cpu_percent) AS 'Average CPU Utilization In Percent',
    MAX(avg_cpu_percent) AS 'Maximum CPU Utilization In Percent',
    AVG(avg_data_io_percent) AS 'Average Data IO In Percent',
    MAX(avg_data_io_percent) AS 'Maximum Data IO In Percent',
    AVG(avg_log_write_percent) AS 'Average Log Write I/O Throughput Utilization In Percent',
    MAX(avg_log_write_percent) AS 'Maximum Log Write I/O Throughput Utilization In Percent' ,
    max(storage_in_megabytes) AS 'Maximum storage size in megabytes',
    MAX(allocated_storage_in_megabytes) AS 'Allocated storage size in megabytes'
FROM sys.resource_stats rs inner join sys.databases db on rs.database_name = db.name
where sku = 'DW' GROUP BY database_name

```

2.9 Total I/O Usage by Database

This query provides the total I/O usage by databases. Execute it in master database and save the results in CSV format.

```
-- Total I/O Usage by Database
--should be execute in master
SELECT Name AS Database_Name
,SUM(num_of_reads)AS Number_of_Reads
,SUM(num_of_writes)AS Number_of_Writes
,(SUM(num_of_reads) + SUM(num_of_writes)) as TotalIO
FROM sys.dm_io_virtual_file_stats(NULL,NULL) I
INNER JOIN sys.databases D ON I.database_id = d.database_id
GROUP BY name
ORDER BY TotalIO desc
```

2.10 Top CPU and I/O Consuming Queries

This query provides top CPU and I/O consuming queries. Execute it a Synapse database and save the results in CSV format.

```
--CPU and I/O Consuming queries using system tables having limit of 10K rows.
--To be executed on Synapse DB
set nocount on;
DECLARE @startdt VARCHAR(10);
DECLARE @enddt VARCHAR(10);

-- Provide start and end date
SET @startdt = '2017-03-05'
SET @enddt = '2023-11-23'

SELECT
concat('IDWWM', '~',
CURRENT_TIMESTAMP, '~',
,s.session_id, '~',
,r.request_id, '~',
,r.status, '~',
,isnull(cast(r.resource_allocation_percentage as nvarchar(20)), ''), '~',
,s.login_name, '~')
```

```

,r.submit_time,'~'
,r.end_compile_time,'~'
,r.total_elapsed_time,'~'
,isnull(ISNULL(r.command2, r.command),'') , '~'
,isnull(sum(ner.cpu_time)/1000,'') , '~'
,isnull(sum(ner.reads),'') , '~'
,isnull(sum(ner.writes),'') , '~'
,isnull(max(ner.granted_query_memory),'') , '~'
,isnull(round(log(sum(g.query_cost)),2,''))

FROM sys.dm_pdw_exec_requests r
JOIN sys.dm_pdw_exec_sessions s
    on r.session_id = s.session_id
LEFT JOIN sys.dm_pdw_request_steps rs
    ON r.request_id = rs.request_id
LEFT JOIN sys.dm_pdw_sql_requests sr
    ON rs.request_id = sr.request_id
    AND rs.step_index = sr.step_index
Left JOIN sys.dm_pdw_nodes_exec_requests ner
    ON sr.spid = ner.session_id
    AND sr.pdw_node_id = ner.pdw_node_id
LEFT JOIN sys.dm_pdw_nodes_exec_query_memory_grants g
    ON ner.session_id = g.session_id
    AND ner.pdw_node_id = g.pdw_node_id
WHERE      cast(r.submit_time as date) BETWEEN CAST(@startdt+' 00:00:00:00' as DATETIME) AND
CAST(@enddt+' 00:00:00:00' AS DATETIME)
GROUP BY r.request_id
        ,r.request_id
        ,r.status
        ,r.command
        ,r.command2
        ,r.resource_allocation_percentage
        ,s.session_id
        ,s.login_name
        ,r.submit_time
        ,r.end_compile_time
        ,r.total_elapsed_time
        ,r.[label]
        ,r.classifier_name
        ,r.group_name
        ,r.result_cache_hit;

```

```
-- CPU and I/O Consuming queries using Data store Tables

set nocount on;
DECLARE @startdt VARCHAR(10)
DECLARE @enddt VARCHAR(10)

-- Provide start and end date
SET @startdt = '2013-03-05'
SET @enddt = '2023-11-23'

SELECT concat('IDWWM','~~',
CURRENT_TIMESTAMP,'~~',
q.query_id,'~~',
qt.query_text_id,'~~'
,isnull(rs.avg_physical_io_reads,''),'~~'
,isnull(rs.avg_cpu_time,''),'~~'
,p.plan_id,'~~'
,isnull(rs.runtime_stats_id,''),'~~'
,qt.query_sql_text , '~~'
,rsi.start_time, '~~'
,rsi.end_time, '~~'
,rs.avg_rowcount, '~~'
,rs.count_executions )
FROM sys.query_store_query_text AS qt
JOIN sys.query_store_query AS q
    ON qt.query_text_id = q.query_text_id
JOIN sys.query_store_plan AS p
    ON q.query_id = p.query_id
JOIN sys.query_store_runtime_stats AS rs
    ON p.plan_id = rs.plan_id
JOIN sys.query_store_runtime_stats_interval AS rsi
    ON rsi.runtime_stats_interval_id = rs.runtime_stats_interval_id
WHERE rsi.start_time >= CAST(@startdt+' 00:00:00:00' as DATETIME) and rsi.end_time <=
CAST(@enddt+' 00:00:00:00' AS DATETIME)
ORDER BY rs.avg_physical_io_reads DESC;
```

Change the value of start date and end date in the parameter at the beginning of the script. Save the output of above query in CSV format if you are executing it from an editor such as SSMS. If the log size is too large, then use the alternative method (see below) to generate the output.



SQL_Server_Query_log_1.sql (Command Line)



SQL_Server_Query_log_2.sql (Command Line)

Copy the attached SQL script (SQL_Server_query_log_1.sql and SQL_Server_query_log_2.sql) at an appropriate location. Change **start** and **end** date as required. Execute it using the below command from Windows command line and share the output file.

This method can be used to execute any SQL command through the command prompt.

```
sqlcmd -S server -U username -P password -d database -I -i ./SQL_Server_query_log_1.sql -y0 -o
output_file_1.log
```

```
sqlcmd -S server -U username -P password -d database -I -i ./SQL_Server_query_log_2.sql -y0 -o
output_file_2.log
```

2.11 Top CPU and I/O Consuming Procedures

This query provides top CPU and I/O consuming procedures.

```
DECLARE @startdt VARCHAR(10)
DECLARE @enddt VARCHAR(10)

SET @startdt = '2019-03-05'
SET @enddt = '2023-04-10'

select CONCAT('IDWWMM', '~',
            sess.login_name, '~',
            coalesce(sess.client_id, ''), '~',
            sess.app_name, '~',
            convert(VARCHAR, d.last_execution_time, 120), '~',
            (d.total_elapsed_time/d.execution_count)/1000000.0, '~',
            (total_logical_reads+total_physical_reads+total_logical_writes)/d.execution_count, '~',
            0, '~',
            convert(VARCHAR, (d.last_execution_time + d.total_elapsed_time/86400000000), 120), '~',
            convert(VARCHAR, d.last_execution_time, 120), '~',
            0, '~',
            row_number() over (ORDER BY total_worker_time/d.execution_count desc), '~',
            (total_worker_time/execution_count)/1000000.0, '~',
            (total_logical_reads+total_physical_reads+total_logical_writes), '~',
            tb4.ampcputime, '~',
            tb4.TotalIOCount, '~',
            d.total_elapsed_time/1000000.0, '~',
            OBJECT_SCHEMA_NAME(d.object_id, d.database_id), '~')
```

```

        db.name, '~',
        REPLACE(CAST(text as NVARCHAR(MAX)), CHAR(10), ' '), '~',
        execution_count
    )
FROM sys.dm_pdw_nodes_exec_procedure_stats AS d
LEFT OUTER JOIN sys.dm_pdw_nodes_exec_sql_text st on (d.sql_handle = st.sql_handle)
INNER JOIN
(
    SELECT DISTINCT
    a.last_execution_time,
    SUM(total_worker_time)/1000000.0 as AMPCPUTIME,
    (SUM(total_logical_reads)+sum(total_physical_reads)+sum(total_logical_writes)) AS TOTALIOCOUNT
    FROM
    sys.dm_pdw_nodes_exec_procedure_stats AS a
    WHERE
    cast(a.last_execution_time as date) BETWEEN CAST(@startdt+' 00:00:00:00' as DATETIME) AND
    CAST(@enddt+' 00:00:00:00' AS DATETIME)
    GROUP BY a.last_execution_time
)TB4 ON TB4.last_execution_time = d.last_execution_time
inner join sys.databases db
on ( db.database_id = d.database_id)
LEFT JOIN sys.dm_pdw_exec_sessions AS Sess
ON (Sess.session_id) = 'SID'+cast(st.session_id as nvarchar(60));

```

2.12 Data for Partitioning and Bucketing

This query extracts table size and column details to be utilized for partition and bucket recommendation. Execute it and save the results in CSV format.

Note: This query provides the details for the database that we are in. You need to execute it for all the databases. Also, you need to generate the statistics before executing this query.

```

WITH base
AS
(
    SELECT
    GETDATE() AS [execution_time]
    , DB_NAME() AS [database_name]
    , s.name AS [schema_name]
    , t.name AS [table_name]
    , QUOTENAME(s.name)+'.'+QUOTENAME(t.name) AS [two_part_name]
    , nt.[name] AS [node_table_name]

```

```

, ROW_NUMBER() OVER(PARTITION BY nt.[name] ORDER BY (SELECT NULL)) AS
[node_table_name_seq]
, tp.[distribution_policy_desc] AS [distribution_policy_name]
, c.[name] AS [distribution_column]
, nt.[distribution_id] AS [distribution_id]
, i.[type] AS [index_type]
, i.[type_desc] AS [index_type_desc]
, nt.[pdw_node_id] AS [pdw_node_id]
, pn.[type] AS [pdw_node_type]
, pn.[name] AS [pdw_node_name]
, di.name AS [dist_name]
, di.position AS [dist_position]
, nps.[partition_number] AS [partition_nmbr]
, nps.[reserved_page_count] AS [reserved_space_page_count]
, nps.[reserved_page_count] - nps.[used_page_count] AS [unused_space_page_count]
, nps.[in_row_data_page_count]
+ nps.[row_overflow_used_page_count]
+ nps.[lob_used_page_count] AS [data_space_page_count]
, nps.[reserved_page_count]
- (nps.[reserved_page_count] - nps.[used_page_count])
- ([in_row_data_page_count]
+ [row_overflow_used_page_count] + [lob_used_page_count]) AS [index_space_page_count]
, nps.[row_count] AS [row_count]
from
sys.schemas s
INNER JOIN sys.tables t
ON s.[schema_id] = t.[schema_id]
INNER JOIN sys.indexes i
ON t.[object_id] = i.[object_id]
AND i.[index_id] <= 1
INNER JOIN sys.pdw_table_distribution_properties tp
ON t.[object_id] = tp.[object_id]
INNER JOIN sys.pdw_table_mappings tm
ON t.[object_id] = tm.[object_id]
INNER JOIN sys.pdw_nodes_tables nt
ON tm.[physical_name] = nt.[name]
INNER JOIN sys.dm_pdw_nodes pn
ON nt.[pdw_node_id] = pn.[pdw_node_id]
INNER JOIN sys.pdw_distributions di
ON nt.[distribution_id] = di.[distribution_id]
INNER JOIN sys.dm_pdw_nodes_db_partition_stats nps
ON nt.[object_id] = nps.[object_id]

```

```

    AND nt.[pdw_node_id] = nps.[pdw_node_id]
    AND nt.[distribution_id] = nps.[distribution_id]
LEFT OUTER JOIN (select * from sys.pdw_column_distribution_properties ) cdp
    ON t.[object_id] = cdp.[object_id]
LEFT OUTER JOIN sys.columns c
    ON cdp.[object_id] = c.[object_id]
    AND cdp.[column_id] = c.[column_id]
WHERE pn.[type] = 'COMPUTE'
)
, size
AS
(
SELECT
    [execution_time]
, [database_name]
, [schema_name]
, [table_name]
, [two_part_name]
, [node_table_name]
, [node_table_name_seq]
, [distribution_policy_name]
, [distribution_column]
, [distribution_id]
, [index_type]
, [index_type_desc]
, [pdw_node_id]
, [pdw_node_type]
, [pdw_node_name]
, [dist_name]
, [dist_position]
, [partition_nmbr]
, [reserved_space_page_count]
, [unused_space_page_count]
, [data_space_page_count]
, [index_space_page_count]
, [row_count]
, ([reserved_space_page_count] * 8.0) AS [reserved_space_KB]
, ([reserved_space_page_count] * 8.0)/1000 AS [reserved_space_MB]
, ([reserved_space_page_count] * 8.0)/1000000 AS [reserved_space_GB]
, ([reserved_space_page_count] * 8.0)/1000000000 AS [reserved_space_TB]
, ([unused_space_page_count] * 8.0) AS [unused_space_KB]
, ([unused_space_page_count] * 8.0)/1000 AS [unused_space_MB]

```



```

, ([unused_space_page_count] * 8.0)/1000000 AS [unused_space_GB]
, ([unused_space_page_count] * 8.0)/1000000000 AS [unused_space_TB]
, ([data_space_page_count] * 8.0) AS [data_space_KB]
, ([data_space_page_count] * 8.0)/1000 AS [data_space_MB]
, ([data_space_page_count] * 8.0)/1000000 AS [data_space_GB]
, ([data_space_page_count] * 8.0)/1000000000 AS [data_space_TB]
, ([index_space_page_count] * 8.0) AS [index_space_KB]
, ([index_space_page_count] * 8.0)/1000 AS [index_space_MB]
, ([index_space_page_count] * 8.0)/1000000 AS [index_space_GB]
, ([index_space_page_count] * 8.0)/1000000000 AS [index_space_TB]
FROM base
)
SELECT
execution_time,database_name,schema_name,table_name,distribution_column,sum(row_count),sum(data_space_gb)
from size group by execution_time,database_name,schema_name,table_name,distribution_column

```

3. Source Code Assessment

Provide the below active or in-scope Synapse source code artifacts as applicable in the migration scope.

Sl. No.	Code Artifact	Criticality	Remarks
1	Orchestration Scripts (Control-M / AutoSys / Cron etc.)	Must	To identify interdependencies across scheduler scripts / jobs, queries, and dependent workloads
2	Procedures / Functions	Must	To identify workload complexity, query patterns, query count etc. for effort estimation and technical debt
3	Views	Must	To identify view complexity, patterns and effort estimations
4	Shell Scripts	Must	To identify count, dependencies, SQL queries and PL/SQL, logic (example: email, notification etc.) and effort estimations
5	DDL	Must	To identify column usage, and provide recommendation on column level lineage, and query optimization on the target system
6	DML / SQL Files	Must	To identify count, dependencies, SQL queries and effort estimations

Note:

Limit: Assuming the orchestration script is a trigger point for every single use case execution in the existing setup. If the customer is not comfortable sharing all the workloads, then share those workloads which are referred or executed through the orchestration scripts. However, in such scenarios the scope and effort estimates will be based on the given workloads.