

# SQL Server Query Execution Logs and Source Code Extraction Prerequisites

## Contents

1. Introduction .....	3
2. Artifact Extraction Summary .....	3
3. Artifact Extraction Steps .....	4
3.1 SQL Server Source Code (Stored Procedures, Functions, Views) .....	4
3.2 Shell Scripts (KSH or Any other batch files).....	5
3.3 DDLs.....	5
3.4 Orchestration – AutoSys .....	6
3.5 SSIS Package extraction.....	6
3.6 Query logs .....	<b>Error! Bookmark not defined.</b>
3.7 Databases .....	8
3.7.1 Databases.....	8
3.7.2 Database Volume .....	8
3.7.3 High Data Volume Tables.....	9
3.7.4 Users .....	11
3.7.5 User Tables.....	11
3.7.6 Objects .....	12
3.7.7 Total I/O Usage by Database .....	12
3.7.8 Top CPU and I/O Consuming Queries .....	13
3.7.9 Top CPU and I/O Consuming Procedures .....	14
3.7.10 Additional Database Objects .....	16
3.7.11 Top CPU and I/O Consuming Functions.....	18
3.7.12 Data for Partitioning and Bucketing .....	20
4. Getting Help.....	22

## 1. Introduction

LeapLogic Assessment profiles existing inventory, identifies workload complexity, performs dependency analysis, and provides recommendations for migration to the modern data platform.

## 2. Artifact Extraction Summary

LeapLogic requires certain artifacts to perform an assessment. All these artifacts should preferably be in the form of .sql files.

Sl. No.	Artifacts needed	Steps	Priority	Description
1	Source code - Procedures and Functions	See <a href="#">here</a>	Must have	To assess the actual source code for complexity To identify workload complexity, query patterns, query count etc. for effort estimation and technical debt
2	Source Code - Views	See <a href="#">here</a>	Must have	To identify view complexity, patterns, and effort estimations
3	Database Details	See <a href="#">here</a>	Must have	To identify database level volume & additional details
4	Shell Scripts	See <a href="#">here</a>	Must have	To identify count, dependencies, SQL queries and PL/SQL, logic (example: email, notification etc.) and estimate effort
5	DDL	See <a href="#">here</a>	Must have	To identify column usage, provide recommendation on column level lineage, and optimize query on the target system
6	Orchestration - AutoSys jobs	See <a href="#">here</a>	Good to have	Helps with orchestration schedule and with removing any non-running jobs. Assesses ad hoc runs.
7	SSIS Package Extraction	See <a href="#">here</a>	Must have	LeapLogic Assessment profiles existing inventory, identifies complexity, draws lineage and provides comprehensive recommendations for migration to modern data platform.
8	Query logs	See <a href="#">here</a>	Good to have	If source code is not given then query logs will help in the overall assessment of workloads and queries

### 3. Artifact Extraction Steps

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

#### 3.1 SQL Server Source Code (Stored Procedures, Functions, Views)

Below code snippet can be used to extract Source code & DDLs for Procedures, Views, Functions

For detailed steps, please refer to the [Additional Database Objects](#) section.

```
-----Read Me-----
-- This script will generate a single file for all databases

-- Modify outfilepath based on your path
-- Filters can be applied based on requirement

--How to run in SSMS
--  Query --> SQLCMDMODE
--  Query --> Results To --> Results To Text
--  Query --> Query Options --> Results --> Text --> Max column chars 1000000
(to a number to fit all code)

DECLARE @DatabaseName NVARCHAR(128)
DECLARE @SQL NVARCHAR(MAX)
SET NOCOUNT ON;
SET ANSI_WARNINGS OFF;
--Out File path
:setvar outfilepath "Test.txt"

-- Create a cursor to loop through databases
DECLARE db_cursor CURSOR FOR
SELECT name
FROM sys.databases
WHERE NAME NOT IN ('master','tempdb','model','msdb','SQLAudit','DBA','DBAdmin')
-- Filter unwanted databases

-- Initialize the cursor
OPEN db_cursor
FETCH NEXT FROM db_cursor INTO @DatabaseName

-- Loop through each database
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Create dynamic SQL to switch database context and retrieve source code
    SET @SQL = N'
        USE [' + @DatabaseName + N'];
        select
```

```

concat(
  'IDW_WM ~~'',
  db_name() collate SQL_Latin1_General_CP1_CI_AS,
  '_ ',
  s.name collate SQL_Latin1_General_CP1_CI_AS,
  '_ ',
  ao.name collate SQL_Latin1_General_CP1_CI_AS,
  '~~'',
  ao.type_desc collate SQL_Latin1_General_CP1_CI_AS,
  '~~'',
  definition collate SQL_Latin1_General_CP1_CI_AS
)
from
sys.all_sql_modules asm
join sys.all_objects ao on
asm.object_id = ao.object_id
join sys.schemas s on
ao.schema_id = s.schema_id
where
ao.is_ms_shipped <>1;
'

-- set out file path
:OUT $(outfilepath)

      -- Execute the dynamic SQL
      EXEC sp_executesql @SQL
      FETCH NEXT FROM db_cursor INTO @DatabaseName
END

-- Clean up
CLOSE db_cursor
DEALLOCATE db_cursor

```

### 3.2 Shell Scripts (KSH or Any other batch files)

Client to provide applicable shell scripts from the production repository.

### 3.3 DDLs

Client DBAs need to extract the DDLs from the production instance for all applicable databases.

### 3.4 Orchestration – AutoSys

LeapLogic requires Autosys jobs exported in .jil/.txt format which typically contains all the boxes, jobs, FW, etc. Follow the steps given below to export the jobs either through CLI or UI.

#### Using CLI

- Connect server to access the AutoSys CLI.
- Execute `"autorep -J % -q > {file path} &"` command on the AutoSys CLI
  - **{file path}**: full path (path + name of file) to save output into file.
  - **Example:** `autorep -J % -q > /tmp/my_job_details.txt &`

#### Using UI

- Open AutoSys job monitoring UI.
- Using the top navigation bar, navigate to **File > Save View as JIL**.
- Save the output as a file.

#### Important Notes:

- It may take several minutes to generate the output (depending upon the number of jobs)
- Limit: Assuming the orchestration script is a trigger point for every single use case execution in the existing setup. If you are not comfortable sharing all the workloads, then share those workloads which are referred to or executed through the orchestration scripts. However, in such scenarios the scope and effort estimates will be based on the given workloads.

### 3.5 SSIS Package extraction

LeapLogic requires SSIS packages to perform an assessment. Follow the below given steps to achieve the same.

1. Use the below path where SSIS packages are stored. Usually, this location is configured when a new project is created. Refer location in the below screenshot.

# Configure your new project

## Integration Services Project

Project name

Integration Services Project1

Location

C:\Users

Solution

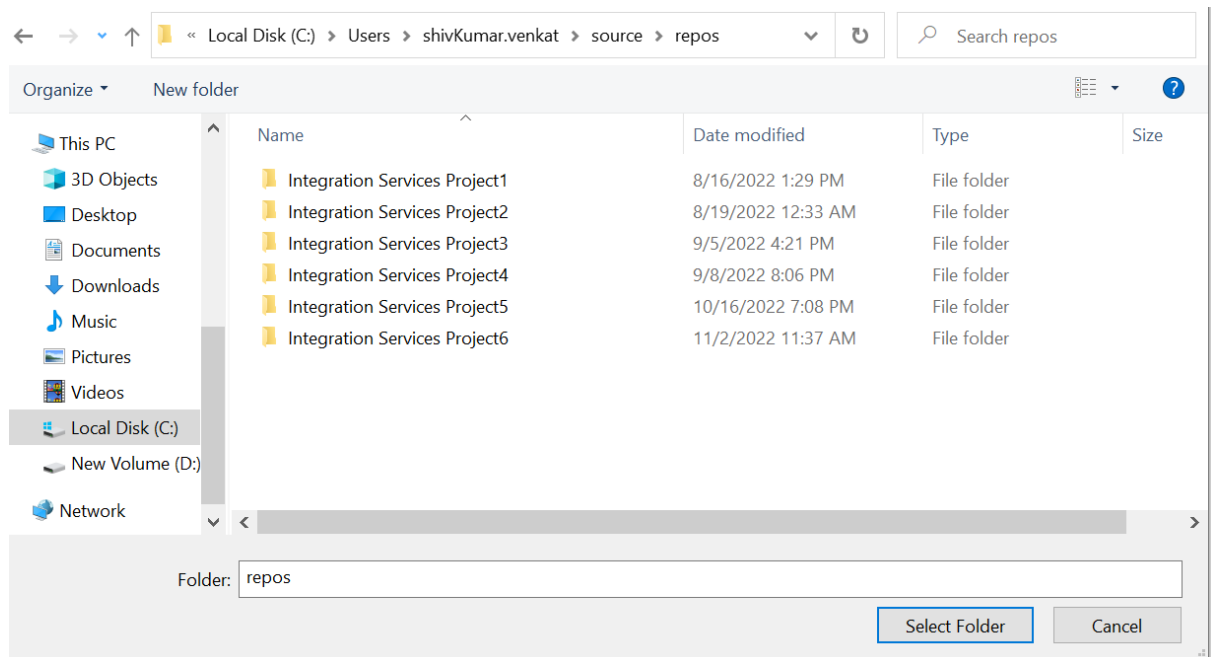
Create new solution

Solution name ⓘ

Integration Services Project1

☐ Place solution and project in the same directory

2. Go to the path C:\Users\<SystemName>\source\repos. You'll see all SSIS packages at this location.



3. Archive the extracted packages and share them

## 3.6 Query logs

Please run the below query and extract the query log in text format.

```

Query#1:
select
concat(
'<LLR>',
format(creation_time,'dd-MM-yyyy'),'<LLC>',
qs.total_elapsed_time/1000,'<LLC>',
db_name(st.dbid) ,'<LLC>',
object_schema_name(st.objectid, st.dbid)+'.'+object_name(st.objectid,
st.dbid),'<LLC>',
st.text, '<LLC>'
)as query_log
from sys.dm_exec_query_stats as qs
cross apply sys.dm_exec_sql_text(qs.[sql_handle]) as st
Query#2
select
concat(
'<LLR>',
spid,'<LLC>',
StartTime,'<LLC>',
[RunTime(m)], '<LLC>',
Login,name,'<LLC>',
GroupNm,'<LLC>',
host,'<LLC>',
BlkBy,'<LLC>',
DBName,'<LLC>',
SQLStatement,'<LLC>',
ObjectName,'<LLC>',
ClientAddress, '<LLC>'
)
from
Util.dbo.DBA_QueryAudit where StartTime >= GETDATE()-90;

```

## 3.7 Databases

### 3.7.1 Databases

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

```
select count (*) from sys.databases
```

### 3.7.2 Database Volume

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



```

DECLARE @command varchar(4000)
SELECT @command = '
USE [?]
SELECT
database_name = CAST(DB_NAME(database_id) AS VARCHAR(50))
, log_size_mb = CAST(SUM(CASE WHEN type_desc = "LOG" THEN size END) * 8. / 1024
AS DECIMAL(8,2))
, row_size_mb = CAST(SUM(CASE WHEN type_desc = "ROWS" THEN size END) * 8. /
1024 AS DECIMAL(8,2))
, total_size_mb = CAST(SUM(size) * 8. / 1024 AS DECIMAL(8,2))
FROM sys.master_files WITH(NOWAIT)
WHERE database_id = DB_ID()
GROUP BY database_id
'
DECLARE @Databases TABLE
(
    database_name VARCHAR(50),
    log_size_mb DECIMAL(8,2),
    row_size_mb DECIMAL(8,2),
    total_size DECIMAL(8,2)
)
INSERT INTO @Databases
EXEC sp_MSforeachdb @command
select * FROM @Databases

```

name	db_size
OWConfiguration	16.00 MB
OWDiagnostics	1072.00 MB
OWQueue	16.00 MB
ep	16.00 MB
plus	16.00 MB
naster	7.38 MB
nodel	16.00 MB
nsdb	46.75 MB
ef	16.00 MB
ef1	16.00 MB
ReportServer	80.00 MB
ReportServerTempDB	16.00 MB
amndh	40.00 MB

### 3.7.3 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.

```

DECLARE @command varchar(4000)
SELECT @command = '
USE [?]
select

```

```

database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
t.NAME AS TableName,
s.Name AS SchemaName,
p.rows AS RowCounts,
CAST(ROUND(((SUM(a.total_pages) * 8) / 1024.00/1024.00), 2) AS NUMERIC(36, 2))
AS TotalSpaceGB,
CAST(ROUND(((SUM(a.used_pages) * 8) / 1024.00/1024.00), 2) AS NUMERIC(36, 2))
AS UsedSpaceGB,
CAST(ROUND(((SUM(a.total_pages) - SUM(a.used_pages)) * 8) / 1024.00/1024.00, 2)
AS NUMERIC(36, 2)) AS UnusedSpaceGB
FROM
    sys.tables t
INNER JOIN
    sys.indexes i ON t.OBJECT_ID = i.object_id
INNER JOIN
    sys.partitions p ON i.object_id = p.OBJECT_ID AND i.index_id = p.index_id
INNER JOIN
    sys.allocation_units a ON p.partition_id = a.container_id
LEFT OUTER JOIN
    sys.schemas s ON t.schema_id = s.schema_id
WHERE
    t.is_ms_shipped = 0
    AND i.OBJECT_ID > 255
GROUP BY
    t.Name, s.Name, p.Rows
HAVING CAST(ROUND(((SUM(a.used_pages) * 8) / 1024.00/1024.00), 2) AS
NUMERIC(36, 2)) >=10
ORDER BY      UsedSpaceGB'
DECLARE @UserTable TABLE
(
    database_name VARCHAR(100),
    Table_Name VARCHAR(500),
    Scheme_Name VARCHAR(50),
    Row_Count BIGINT,
    Space_Total_GB DECIMAL(10,2),
    Space_Used_GB DECIMAL(10,2),
    Space_UnUsed_GB DECIMAL(10,2)
)
INSERT INTO @UserTable
EXEC sp_MSforeachdb @command
select * FROM @UserTable

```

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

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

### 3.7.4 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.

```
DECLARE @command varchar(4000)
SELECT @command = '
USE [?]
select
database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
type ,count(*)
from sys.database_principals
group by type'
DECLARE @UserType TABLE
(
    database_name VARCHAR(50),
    UserType VARCHAR(50),
    Num INT
)
INSERT INTO @UserType
EXEC sp_MSforeachdb @command
select * FROM @UserType
```

type

**Type of principal**  
**S = SQL Server user**  
**U = Windows user**  
**G = Windows group**  
**A = Application role**  
**R = Database role**  
**C = Certificate mapped**  
**K = Asymmetric key mapped**

### 3.7.5 User Tables

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

```
DECLARE @command varchar(4000)
SELECT @command = '
USE [?]
select
database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
Name,type_desc,create_date,modify_date
from sys.tables'
DECLARE @UserTable TABLE
(
    database_name VARCHAR(100),
    Table_Name VARCHAR(500),
    Table_Type VARCHAR(50),
```

```

        Create_Date DATETIME,
        ModifiedDate DATETIME

    )

    INSERT INTO @UserTable
    EXEC sp_MSforeachdb @command
    select * FROM @UserTable

```

### 3.7.6 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.

```

DECLARE @command varchar(4000)
SELECT @command = '
USE [?]
select
database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
Name,type_desc,create_date,modify_date
from sys.objects'
DECLARE @UserTable TABLE
(
    database_name VARCHAR(100),
    Obj_Name VARCHAR(500),
    obj_Type VARCHAR(50),
    Create_Date DATETIME,
    ModifiedDate DATETIME
)
INSERT INTO @UserTable
EXEC sp_MSforeachdb @command
select * FROM @UserTable

```

### 3.7.7 Total I/O Usage by Database

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

```

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

```

### 3.7.8 Top CPU and I/O Consuming Queries

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

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

-- Provide start and end date
SET @startdt = '2017-03-05'
SET @enddt = '2022-04-05'

select CONCAT('IDWWM', '~~',
    sess.original_login_name, '~~',
    AppId, '~~',
    sess.client_interface_name, '~~',
    convert(VARCHAR, qs.last_execution_time, 120), '~~',
    (total_worker_time/execution_count)/1000000.0, '~~',
    (total_logical_reads+total_physical_reads+total_logical_writes)/execution_count, '~~',
    0, '~~',
    convert(VARCHAR, (qs.last_execution_time +
    qs.total_elapsed_time/86400000000), 120), '~~',
    convert(VARCHAR, qs.last_execution_time, 120), '~~',
    0, '~~',
    row_number() over (ORDER BY total_worker_time/execution_count desc), '~~',
    (total_worker_time/execution_count)/1000000.0, '~~',
    (total_logical_reads+total_physical_reads+total_logical_writes), '~~',
    tb4.ampcputime, '~~',
    tb4.TotalIOCount, '~~',
    qs.total_elapsed_time/1000000.0, '~~',
    OBJECT_SCHEMA_NAME(st.objectid, st.dbid), '~~',
    db.name, '~~',
    REPLACE(CAST(text as NVARCHAR(MAX)), CHAR(10), ' '), '~~',
    execution_count
    )
    from sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.plan_handle) st
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_exec_query_stats 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 = qs.last_execution_time
inner join sys.databases db
on ( db.database_id = st.dbid)
LEFT JOIN (
SELECT DISTINCT es.program_name as AppId,
es.client_interface_name,es.original_login_name,sp.dbid
FROM sys.dm_exec_sessions es inner join sys.sysprocesses sp
on es.session_id = sp.spid
) AS Sess
ON Sess.dbid = st.dbid;

```

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.sql

Copy the attached SQL script (SQL\_Server\_query\_log.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 SQL_Server_Name -U username -P password -i ./SQL_Server_query_log.sql
-y0 -o output_file.log

```

### 3.7.9 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)

-- Provide start and end date
SET @startdt = '2024-03-05'
SET @enddt = '2024-09-05'

select CONCAT('IDWWM', '~',
    sess.original_login_name, '~',
    AppId, '~',
    sess.client_interface_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_exec_procedure_stats AS d
CROSS APPLY sys.dm_exec_sql_text(d.sql_handle) st
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_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 (
    SELECT DISTINCT es.program_name as AppId,
    es.client_interface_name,es.original_login_name,sp.dbid
    FROM sys.dm_exec_sessions es inner join sys.sysprocesses sp
    on es.session_id = sp.spid
) AS Sess
ON Sess.dbid = d.database_id;

```

### 3.7.10 Additional Database Objects

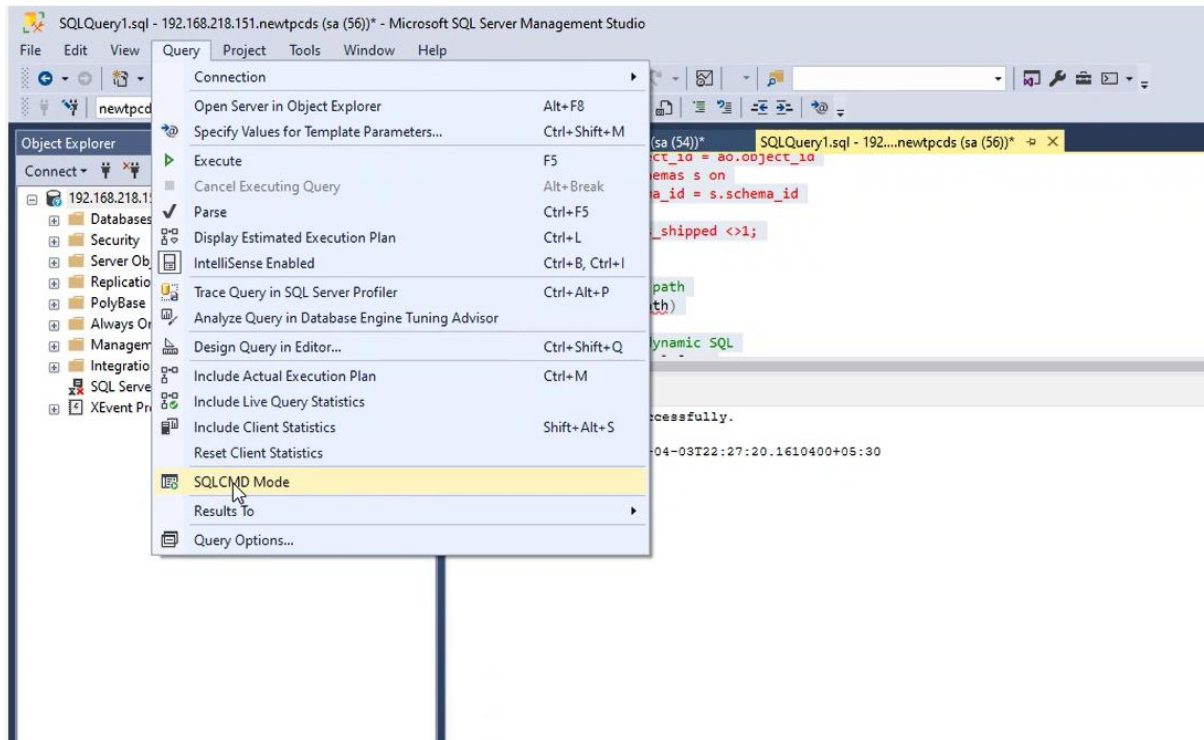
Export additional database objects such as stored procedures, views, etc. using the below attached script.



Script\_to\_extract\_SQL  
Server\_Object\_definiti

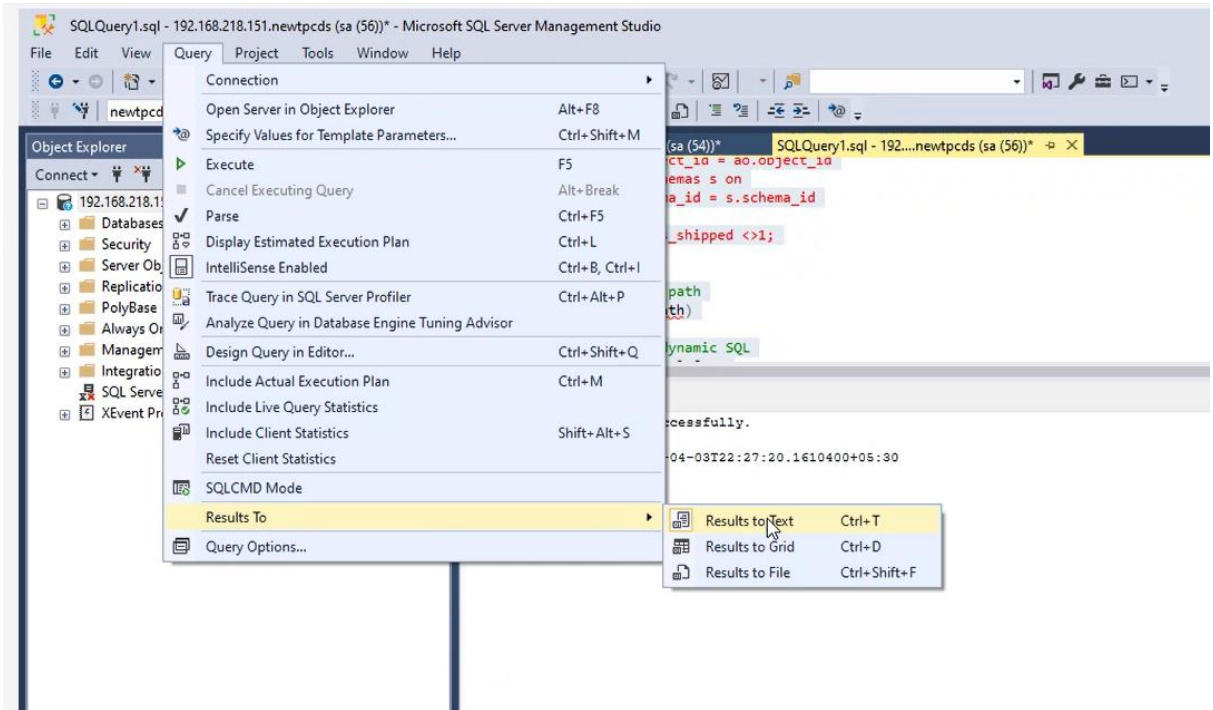
The steps to execute this script using the SQL Server Management Studio (SSMS) tool are as follows.

1. Navigate to **Query > SQL CMD Mode**

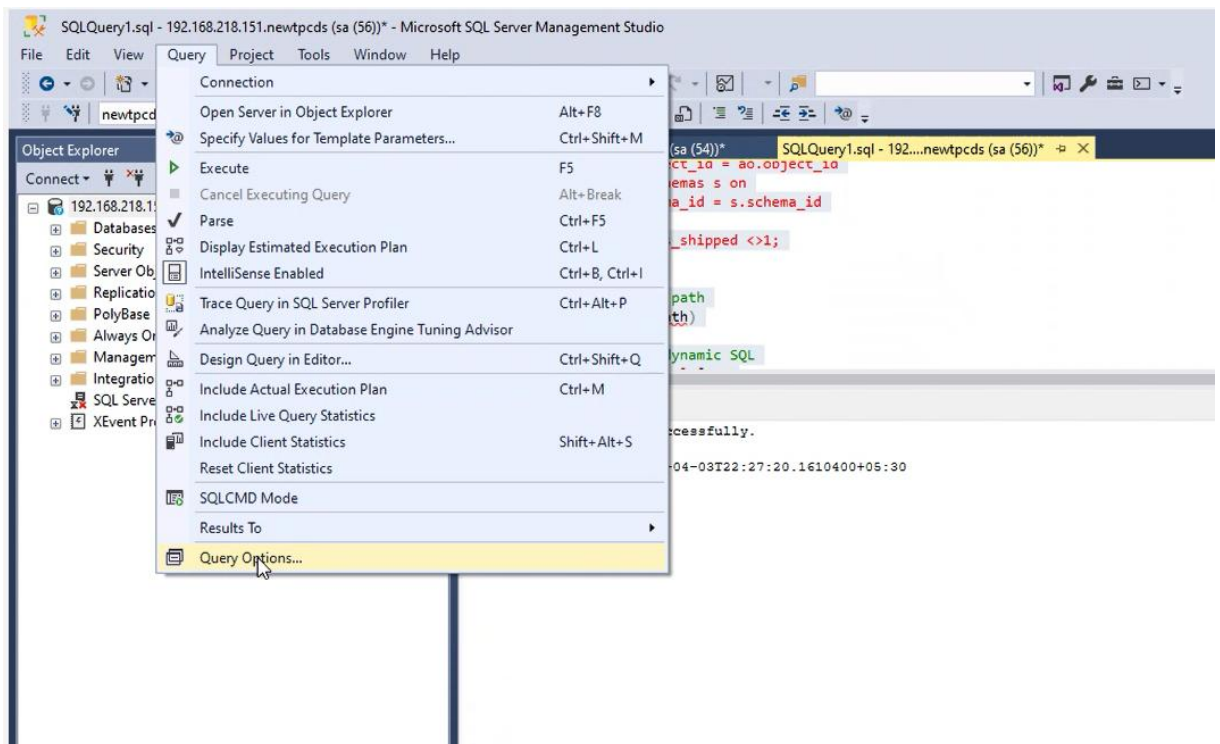


2. Next, configure the results using **Query > Results To > Results to Text**

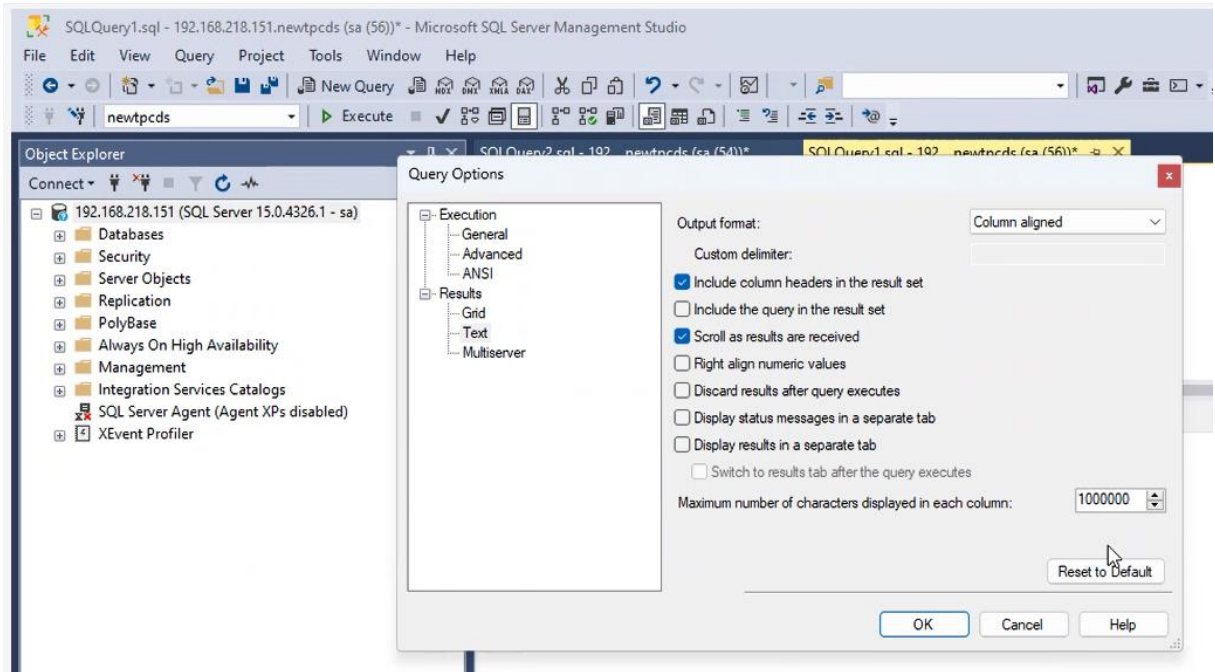




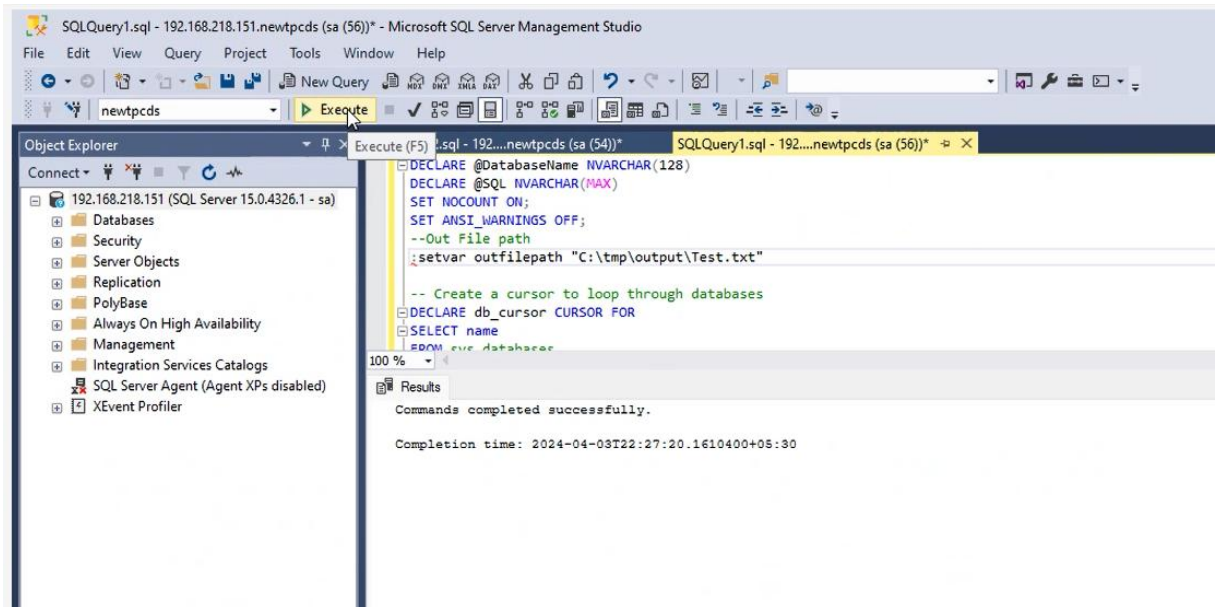
3. Now, navigate to **Query > Query Options**



4. Configure Results > Text > Maximum number of characters displayed in each: 1000000 (to a number to fit all code)



**Note:** Please set the path in the script before executing it :setvar outfilepath "Test.txt"



### 3.7.11 Top CPU and I/O Consuming Functions

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

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

-- Provide start and end date
SET @startdt = '2017-03-05'
SET @enddt = '2022-04-05'
```

```

select CONCAT('IDWWM', '~',
    sess.original_login_name, '~',
    AppId, '~',
    sess.client_interface_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/864000000000),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_exec_function_stats AS d
CROSS APPLY sys.dm_exec_sql_text(d.sql_handle) st
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_exec_function_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 (
SELECT DISTINCT es.program_name as AppId,
es.client_interface_name,es.original_login_name,sp.dbid
FROM sys.dm_exec_sessions es inner join sys.sysprocesses sp
on es.session_id = sp.spid
) AS Sess

```

```
ON Sess.dbid = d.database_id;
```

An alternative method to generate the log file for Queries/Procedures or Functions is using sqlcmd (T-SQL) CLI

### 3.7.12 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.

```
DECLARE @command varchar(4000)
SELECT @command = '
USE [?]
SELECT isc.table_catalog AS DATABASE_NAME,
       t.NAME             AS Table_Name,
       Cast(Round(( Sum(a.used_pages) / 128.00 ), 2) AS NUMERIC(36, 2)) *
       ( 2014 * 1024 )    AS TABLE_SIZE_BYTES,
       p.rows             AS NUM_ROWS,
       isc.column_name    AS Column_Name,
       '''' AS num_unique_val
FROM   sys.tables t
       INNER JOIN sys.indexes i
           ON t.object_id = i.object_id
       INNER JOIN sys.partitions p
           ON i.object_id = p.object_id
           AND i.index_id = p.index_id
       INNER JOIN sys.allocation_units a
           ON p.partition_id = a.container_id
       INNER JOIN sys.schemas s
           ON t.schema_id = s.schema_id
       INNER JOIN information_schema.columns AS isc
           ON t.NAME = isc.table_name
GROUP BY t.NAME,
         s.NAME,
         p.rows,
         isc.table_catalog,
         isc.column_name'
DECLARE @UserTable TABLE
(
    database_name VARCHAR(100),
    Table_Name VARCHAR(500),
    Table_Size_Bytes DECIMAL(18,2),
    Row_Num INT,
```

```
Column_Name VARCHAR(100),  
Unique_Val_Num INT  
)  
INSERT INTO @UserTable  
EXEC sp_MSforeachdb @command  
select * FROM @UserTable
```

#### 4. Getting Help

Contact LeapLogic technical support at [info@leaplogic.io](mailto:info@leaplogic.io)