Working with DMVs and DMFs

Examples using DMVs to monitor statistics on server

1. Monitoring active connections (sessions) on the server

```
SELECT session_id, login_name, host_name, program_name, status
FROM sys.dm_exec_sessions
WHERE status =’running’
```

2. Example DMV to evaluate the slowest performing queries

```
SELECT TOP 5
    total_worker_time/execution_count AS avg_cpu_time,
    text AS query_text
FROM sys.dm_exec_query_stats
CROSS APPLY sys.dm_exec_sql_text(sql_handle)
ORDER BY avg_cpu_time DESC;
```

Explaining the above query

a. SELECT TOP 5: This means we want only the top 5 results from the output, not all rows.
b. total_worker_time/execution_count AS avg_cpu_time:
c. total_worker_time is the total CPU time (in microseconds) used by each query since it was last compiled.
d. execution_count is how many times that query has run.
e. Dividing total_worker_time by execution_count gives the average CPU time per execution, which helps identify queries that use the most CPU on average.
f. SUBSTRING(text, statement_start_offset/2, ...): This part extracts the actual SQL statement text from the full query plan.
g. *text* is column containing the full SQL query whose performance we are evaluating.
h. FROM sys.dm_exec_query_stats: This DMV contains performance statistics for all cached query plans, such as CPU time, execution count, and elapsed time.
i. CROSS APPLY sys.dm_exec_sql_text(sql_handle):
j. This joins the query stats with the actual SQL text for each query, so you can see which query corresponds to which stats.
k. ORDER BY avg_cpu_time DESC:
l. This sorts the results so the queries with the highest average CPU time are shown first, making it easy to spot the most resource-intensive queries.

3. Monitor Physical I/O for Database Files

```
SELECT
    DB_NAME(database_id) AS DatabaseName,
    file_id,
    num_of_reads,
    num_of_writes,
    io_stall_read_ms,
    io_stall_write_ms
FROM sys.dm_io_virtual_file_stats(NULL, NULL);
```

Step-by-Step Explanation

a. sys.dm_io_virtual_file_stats(NULL, NULL): This DMF returns cumulative I/O statistics for all database files. The parameters NULL, NULL mean "all databases and all files".
b. DB_NAME(database_id): Shows the name of the database associated with each file.
c. file_id: Identifies the file within the database.
d. num_of_reads and num_of_writes: Show how many times the file has been read from or written to since the server started.
e. io_stall_read_ms and io_stall_write_ms: Show the total time (in milliseconds) that user processes have waited for read and write operations to complete on each file. High values here may indicate disk bottlenecks.