Efficient Database Monitoring Queries

Commands and Scripts with Examples

Asfaw Gedamu Haileselasie

24/02/2024

This guide outlines efficient methods for querying various database monitoring information in an Oracle database, including commands, scripts, privileges required, and script execution details.

Privileges Required:

Most queries require **SELECT** privilege on relevant tables and views. Some might require additional privileges like **EXECUTE** or **ALTER SESSION**.

Executing Saved Scripts:

1. **Save the script** as a .sql file.
2. **Connect** to your database using a SQL client tool (e.g., SQL*Plus, SQL Developer).
3. **Execute the script** using the appropriate command in your client tool.
4. Script one is general while script two is specific.

1. Find Current Running SQLs:

Script 1:

```
SELECT sid, serial#, sql_id, machine, username, sql_text
FROM v$sql
WHERE state IN ('ACTIVE', 'WAITING');
```

Script 2:

```
select sesion.sid,
sesion.username,
optimizer_mode,
hash_value,
address,
cpu_time,
elapsed_time,
sql_text
from v$sqlarea sqlarea, v$session sesion
where sesion.sql_hash_value = sqlarea.hash_value
and sesion.sql_address = sqlarea.address
and sesion.username is not null;
```

2. Find Active Sessions in Oracle Database:

```
Script 1:


SELECT sid, serial#, username, machine, status
FROM v$session;
```

Script 2:

```
set echo off
set linesize 95
set head on
set feedback on
col sid head "Sid" form 9999 trunc
col serial# form 99999 trunc head "Ser#"
col username form a8 trunc
col osuser form a7 trunc
col machine form a20 trunc head "Client|Machine"
col program form a15 trunc head "Client|Program"
col login form a11
col "last call" form 9999999 trunc head "Last Call|In Secs"
col status form a6 trunc
select sid,serial#,substr(username,1,10)
username,substr(osuser,1,10) osuser,
substr(program||module,1,15) program,substr(machine,1,22)
machine,
to_char(logon_time,'ddMon hh24:mi') login,
last_call_et "last call",status
from gv$session where status='ACTIVE'
order by 1
/
```

3. Find Wait Events in Database:

Script 1:

```
SELECT sid, serial#, event_id, event_name, wait_time, timeouts
FROM v$session_wait;
```

Script 2:

```
SELECT a.sid,substr(b.username,1,10)
username,substr(b.osuser,1,10) osuser,
```

```
substr(b.program||b.module,1,15) program,substr(b.machine,1,22)
machine,
a.event,a.p1,b.sql_hash_value
from v$session_wait a,V$session b
where b.sid=a.sid
and a.event not in('SQL*Net message from client','SQL*Net
message to client',
'smon timer','pmon timer')
and username is not null
order by 6
/
```

4. Find Sessions Generating Undo:

Script 1:

```
SELECT s.sid, s.serial#, s.username, s.machine, u.undo_blks
FROM v$session s
INNER JOIN v$session_undo u ON s.sid = u.sid;
```

Script 2:

```
select a.sid, a.serial#, a.username, b.used_urec
used_undo_record, b.used_ublk used_undo_blocks
from v$session a, v$transaction b
where a.saddr=b.ses_addr ;
```

5. Find the Temp Usage of Sessions:

Script 1:

```
SELECT sid, serial#, username, machine, temp_used
FROM v$session;
```

Script 2:

```
SELECT b.tablespace,
ROUND(((b.blocks*p.value)/1024/1024),2)||'M' AS temp_size,
a.inst_id as Instance,
a.sid||','||a.serial# AS sid_serial,
NVL(a.username, '(oracle)') AS username,
a.program,
a.status,
a.sql_id
```

```
FROM gv$session a,
gv$sort_usage b,
gv$parameter p
WHERE p.name = 'db_block_size'
AND a.saddr = b.session_addr
AND a.inst_id=b.inst_id
AND a.inst_id=p.inst_id
ORDER BY temp_size desc
/
```

6. Find Sessions Generating Lot of Redo:

Script 1:

```
SELECT s.sid, s.serial#, s.username, s.machine, r.redo_size
FROM v$session s
INNER JOIN v$session_longops r ON s.sid = r.sid
WHERE operation = 'SQL*Net message from client';
```

Script 2:

```
set lines 2000
set pages 1000
col sid for 99999
col name for a09
col username for a14
col PROGRAM for a21
col MODULE for a25
select s.sid,sn.SERIAL#,n.name, round(value/1024/1024,2)
redo_mb, sn.username,sn.status,substr (sn.program,1,21)
"program", sn.type, sn.module,sn.sql_id
from v$sesstat s join v$statname n on n.statistic# =
s.statistic#
join v$session sn on sn.sid = s.sid where n.name like 'redo
size' and s.value!=0 order by
redo_mb desc;
```

7. Get Size of the Database:

Script 1:

```
SELECT SUM(bytes) / (1024 * 1024) AS size_mb
FROM dba_data_files;
```

Script 2:

```
col "Database Size" format a20
col "Free space" format a20
col "Used space" format a20
select round(sum(used.bytes) / 1024 / 1024 / 1024 ) || ' GB'
"Database Size"
, round(sum(used.bytes) / 1024 / 1024 / 1024 ) -
round(free.p / 1024 / 1024 / 1024) || ' GB' "Used space"
, round(free.p / 1024 / 1024 / 1024) || ' GB' "Free space"
from (select bytes
from v$datafile
union all
select bytes
from v$tempfile
union all
select bytes
from v$log) used
, (select sum(bytes) as p
from dba_free_space) free
group by free.p
/
```

8. Monitor Tablespace Usage:

Script 1:

```
SELECT tablespace_name, SUM(blocks) / (1024 * 1024) AS used_mb
FROM dba_tablespace_usage_stats
GROUP BY tablespace_name;
```

Script 2:

```
set feedback off
set pagesize 70;
set linesize 2000
set head on
COLUMN Tablespace format a25 heading 'Tablespace Name'
COLUMN autoextensible format a11 heading 'AutoExtend'
COLUMN files_in_tablespace format 999 heading 'Files'
COLUMN total_tablespace_space format 99999999 heading
'TotalSpace'
```

```
COLUMN total_used_space format 99999999 heading 'UsedSpace'
COLUMN total_tablespace_free_space format 99999999 heading
'FreeSpace'
COLUMN total_used_pct format 9999 heading '%Used'
COLUMN total_free_pct format 9999 heading '%Free'
COLUMN max_size_of_tablespace format 99999999 heading
'ExtendUpto'
COLUM total_auto_used_pct format 999.99 heading 'Max%Used'
COLUMN total_auto_free_pct format 999.99 heading 'Max%Free'
WITH tbs_auto AS
(SELECT DISTINCT tablespace_name, autoextensible
FROM dba_data_files
WHERE autoextensible = 'YES'),
files AS
(SELECT tablespace_name, COUNT (*) tbs_files,
SUM (BYTES/1024/1024) total_tbs_bytes
FROM dba_data_files
GROUP BY tablespace_name),
fragments AS
(SELECT tablespace_name, COUNT (*) tbs_fragments,
SUM (BYTES)/1024/1024 total_tbs_free_bytes,
MAX (BYTES)/1024/1024 max_free_chunk_bytes
FROM dba_free_space
GROUP BY tablespace_name),
AUTOEXTEND AS
(SELECT tablespace_name, SUM (size_to_grow) total_growth_tbs
FROM (SELECT tablespace_name, SUM (maxbytes)/1024/1024
size_to_grow
FROM dba_data_files
WHERE autoextensible = 'YES'
GROUP BY tablespace_name
UNION
SELECT tablespace_name, SUM (BYTES)/1024/1024 size_to_grow
FROM dba_data_files
WHERE autoextensible = 'NO'
GROUP BY tablespace_name)
GROUP BY tablespace_name)
SELECT c.instance_name,a.tablespace_name Tablespace,
CASE tbs_auto.autoextensible
WHEN 'YES'
THEN 'YES'
ELSE 'NO'
END AS autoextensible,
files.tbs_files files_in_tablespace,
```

```
files.total_tbs_bytes total_tablespace_space,
(files.total_tbs_bytes - fragments.total_tbs_free_bytes
) total_used_space,
fragments.total_tbs_free_bytes total_tablespace_free_space,
round(( ( (files.total_tbs_bytes -
fragments.total_tbs_free_bytes)
/ files.total_tbs_bytes
)
* 100
)) total_used_pct,
round(((fragments.total_tbs_free_bytes / files.total_tbs_bytes)
* 100
)) total_free_pct
FROM dba_tablespaces a,v$instance c , files, fragments,
AUTOEXTEND, tbs_auto
WHERE a.tablespace_name = files.tablespace_name
AND a.tablespace_name = fragments.tablespace_name
AND a.tablespace_name = AUTOEXTEND.tablespace_name
AND a.tablespace_name = tbs_auto.tablespace_name(+)
order by total_free_pct;
```

9. Monitor Undo Tablespace Usage:

Script 1:

```
SELECT tablespace_name, SUM(blocks) / (1024 * 1024) AS used_mb
FROM dba_data_files
WHERE tablespace_name LIKE 'UNDO%';
```

Script 2:

```
select a.tablespace_name, SIZEMB, USAGEMB, (SIZEMB - USAGEMB)
FREEMB
from (select sum(bytes) / 1024 / 1024 SIZEMB, b.tablespace_name
from dba_data_files a, dba_tablespaces b
where a.tablespace_name = b.tablespace_name
and b.contents = 'UNDO'
group by b.tablespace_name) a,
(select c.tablespace_name, sum(bytes) / 1024 / 1024 USAGEMB
from DBA_UNDO_EXTENTS c
where status <> 'EXPIRED'
group by c.tablespace_name) b
where a.tablespace_name = b.tablespace_name;
```

10. Monitor TEMP Tablespace Usage:

Script 1:

```
SELECT tablespace_name, SUM(blocks) / (1024 * 1024) AS used_mb
FROM dba_data_files
WHERE tablespace_name LIKE 'TEMP%';
```

Script 2:

```
select a.tablespace_name tablespace,
d.TEMP_TOTAL_MB,
sum (a.used_blocks * d.block_size) / 1024 / 1024 TEMP_USED_MB,
d.TEMP_TOTAL_MB - sum (a.used_blocks * d.block_size) / 1024 /
1024 TEMP_FREE_MB
from v$sort_segment a,
(
select b.name, c.block_size, sum (c.bytes) / 1024 / 1024
TEMP_TOTAL_MB
from v$tablespace b, v$tempfile c
where b.ts#= c.ts#
group by b.name, c.block_size
) d
where a.tablespace_name = d.name
group by a.tablespace_name, d.TEMP_TOTAL_MB;
```

11. Find Blocking Sessions:

Script 1:

```
SELECT s.sid, s.serial#, s.username, s.machine, s2.sid,
s2.serial#, s2.username, s2.machine
FROM v$session s
JOIN v$session_wait s2 ON s.blocking_session_status = 'ACTIVE'
AND s.sid = s2.blocking_session_sid;
```

Script 2:

```
SELECT
s.inst_id,
s.blocking_session,
s.sid,
s.serial#,
```

```
s.seconds_in_wait
FROM
gv$session s
WHERE
blocking_session IS NOT NULL;
```

12. Find Long Running Operations:

Script 1:

```
SELECT sid, serial#, sql_id, machine, username, elapsed_time
FROM v$sql
WHERE state = 'ACTIVE' AND last_call_et > <threshold_seconds>;
```

Script 2:

```
select
sid,inst_id,opname,totalwork,sofar,start_time,time_remaining
from gv$session_longops
where totalwork<>sofar
/
```

13. Find Locks Present in Database:

Script 1:

```
SELECT owner#, object_type, object_name, session_id, lock_mode,
status
FROM dba_locks;
```

Script 2:

```
col session_id head 'Sid' form 9999
col object_name head "Table|Locked" form a30
col oracle_username head "Oracle|Username" form a10 truncate
col os_user_name head "OS|Username" form a10 truncate
col process head "Client|Process|ID" form 99999999
col mode_held form a15
select lo.session_id,lo.oracle_username,lo.os_user_name,
lo.process,do.object_name,
decode(lo.locked_mode,0, 'None',1, 'Null',2, 'Row Share (SS)',
3, 'Row Excl (SX)',4, 'Share',5, 'Share Row Excl (SSX)',6,
'Exclusive',
to_char(lo.locked_mode)) mode_held
```

```
from v$locked_object lo, dba_objects do
where lo.object_id = do.object_id
order by 1,5
/
```

14. Find Queries Triggered from a Procedure:

Script 1:

```
SELECT name, type, package_name, sql_text
FROM dba_sql_statements
WHERE referenced_owner = '<procedure_owner>'
AND referenced_name = '<procedure_name>';
```

Script 2:

```
--Below script will provide the dependent queries getting
--triggered from a procedure.

SELECT s.sql_id, s.sql_text
FROM gv$sqlarea s JOIN dba_objects o ON s.program_id =
o.object_id
and o.object_name = '&procedure_name';
```

15. Get SID from OS PID:

Script 1:

```
SELECT sid FROM v$process WHERE osuser = <os_pid>;
```

Script 2:

```
set lines 123
col USERNAME for a15
col OSUSER for a8
col MACHINE for a15
col PROGRAM for a20
select b.spid, a.username, a.program , a.osuser ,a.machine,
a.sid, a.serial#, a.status from gv$session a, gv$process b
where addr=paddr(+) and sid=&sid;
```

16. Kill All Sessions of a User:

Script 1:

```
ALTER SESSION SET SQL_TRACE = TRUE;
BEGIN
  DBMS_SESSION.KILL(USERNAME => '<username>');
END;
/
ALTER SESSION SET SQL_TRACE = FALSE;
```

Script 2:

```
BEGIN
FOR r IN (select sid,serial# from v$session where username =
'TEST_ANB')
LOOP
EXECUTE IMMEDIATE 'alter system kill session ''' || r.sid
|| ',' || r.serial# || '''';
END LOOP;
END;
/
```

17. Kill all Sessions of a SQL_ID (Requires DELETE system privilege):

Script 1:

```
SELECT sid FROM v$session WHERE sql_id = '<sql_id>';

FOR rec IN (SELECT * FROM
            (SELECT sid FROM v$session WHERE sql_id =
'<sql_id>') sq)
LOOP
  DBMS_SESSION.KILL(sid => rec.sid);
END LOOP;
/
```

Script 2:

```
select 'alter system kill session ' ||''''||SID||','||SERIAL#||'
immediate ;' from v$session
where sql_id='&sql_id';
--FOR RAC
select 'alter system kill session '
```

```
||''''||SID||',''||SERIAL#||',@''||inst_id||''''||' immediate ;'
from gv$session where sql_id='&sql_id'
```

18. Get Parallel Query Detail (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT * FROM v$sql_plan_monitor WHERE plan_hash_value =
'<plan_hash_value>';
```

Script 2:

```
col username for a9
col sid for a8
set lines 299
select
s.inst_id,
decode(px.qcinst_id,NULL,s.username,
' - '||lower(substr(s.program,length(s.program)-4,4) ) )
"Username",
decode(px.qcinst_id,NULL, 'QC', '(Slave)') "QC/Slave" ,
to_char( px.server_set) "Slave Set",
to_char(s.sid) "SID",
decode(px.qcinst_id, NULL ,to_char(s.sid) ,px.qcsid) "QC SID",
px.req_degree "Requested DOP",
px.degree "Actual DOP", p.spid
from
gv$px_session px,
gv$session s, gv$process p
where
px.sid=s.sid (+) and
px.serial#=s.serial# and
px.inst_id = s.inst_id
and p.inst_id = s.inst_id
and p.addr=s.paddr
order by 5 , 1 desc
/
```

19. Kill Snipped Session in DB (Requires DELETE system privilege):

Script 1:

```
SELECT sid FROM v$session WHERE status = 'SNAPPED';

FOR rec IN (SELECT * FROM
            (SELECT sid FROM v$session WHERE status = 'SNAPPED')
sq)
LOOP
  DBMS_SESSION.KILL(sid => rec.sid);
END LOOP;
/
```

Script 2:

```
--It will generate kill session statements for all snipped
sessions:
select 'alter system kill session '''||sid||','||serial#||'''
immediate;' from v$session where status='SNIPED' ;
```

20. Top Query with High Elapsed Time (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT sql_id, elapsed_time_in_second, executions
FROM v$sql
ORDER BY elapsed_time_in_second DESC
FETCH FIRST 10 ROWS ONLY;
```

Script 2:

```
--Queries in last 1 hour ( Preferred to run from Toad, for
proper view)
Select
module,parsing_schema_name,inst_id,sql_id,CHILD_NUMBER,sql_plan_
baseline,sql_profile,plan_hash_value,sql_fulltext,
to_char(last_active_time,'DD/MM/YY HH24:MI:SS' ),executions,
elapsed_time/executions/1000/1000,
rows_processed,sql_plan_baseline from gv$sql where
last_active_time>sysdate-1/24
and executions <> 0 order by elapsed_time/executions desc
```

21. Monitor Parallel Queries (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT * FROM v$sql_monitor WHERE degree > 1;
```

Script 2:

```
  select
       s.inst_id,
       decode(px.qcinst_id,NULL,s.username,
            ' - '||lower(substr(s.program,length(s.program)-4,4)
) ) "Username",
       decode(px.qcinst_id,NULL, 'QC', '(Slave)') "QC/Slave" ,
       to_char( px.server_set) "Slave Set",
       to_char(s.sid) "SID",
       decode(px.qcinst_id, NULL ,to_char(s.sid) ,px.qcsid) "QC
SID",
       px.req_degree "Requested DOP",
      px.degree "Actual DOP", p.spid
   from
     gv$px_session px,
     gv$session s, gv$process p
   where
     px.sid=s.sid (+) and
     px.serial#=s.serial# and
     px.inst_id = s.inst_id
     and p.inst_id = s.inst_id
     and p.addr=s.paddr
  order by 5 , 1 desc
```

22. Find Locked Objects (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT object_type, object_name, sid, session_type
FROM v$locked_object;
```

Script 2:

```
SET PAGESIZE 1000
SET VERIFY OFF
COLUMN owner FORMAT A20
COLUMN username FORMAT A20
COLUMN object_owner FORMAT A20
```

```
COLUMN object_name FORMAT A30
COLUMN locked_mode FORMAT A15
SELECT b.inst_id,
b.session_id AS sid,
NVL(b.oracle_username, '(oracle)') AS username,
a.owner AS object_owner,
a.object_name,
Decode(b.locked_mode, 0, 'None',
1, 'Null (NULL)',
2, 'Row-S (SS)',
3, 'Row-X (SX)',
4, 'Share (S)',
5, 'S/Row-X (SSX)',
6, 'Exclusive (X)',
b.locked_mode) locked_mode,
b.os_user_name
FROM dba_objects a,
gv$locked_object b
WHERE a.object_id = b.object_id
ORDER BY 1, 2, 3, 4;
SET PAGESIZE 14
SET VERIFY ON
SET VERIFY ON
```

23. Check Open Cursors (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT * FROM v$sql WHERE open_cursors > 0;
```

Script 2:

```
--Current open cursor
select a.value, s.username, s.sid, s.serial#
from v$sesstat a, v$statname b, v$session s
where a.statistic# = b.statistic# and s.sid=a.sid
and b.name = 'opened cursors current';
--Max allowed open cursor and total open cursor
select max(a.value) as highest_open_cur, p.value as max_open_cur
from v$sesstat a, v$statname b, v$parameter p
where a.statistic# = b.statistic# and b.name = 'opened cursors
current'
and p.name= 'open_cursors'
group by p.value;
```

24. Session Login History from ASH (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT * FROM dba_hist_active_sess_history WHERE action='LOGON';
```

Script 2:

```
select c.username,a.SAMPLE_TIME, a.SQL_OPNAME, a.SQL_EXEC_START,
a.program, a.module, a.machine, b.SQL_TEXT
from DBA_HIST_ACTIVE_SESS_HISTORY a, dba_hist_sqltext b,
dba_users c
where a.SQL_ID = b.SQL_ID(+)
and a.user_id=c.user_id
and c.username='&username'
order by a.SQL_EXEC_START asc;
```

25. Buffer Cache Hit Ratio:

Script 1:

```
SELECT 1 - (dbms_buffer_cache.get_miss_ratio) * 100 AS hit_ratio
FROM DUAL;
```

Script 2:

```
SELECT ROUND((1-(phy.value / (cur.value + con.value)))*100,2)
"Cache Hit Ratio"
FROM v$sysstat cur, v$sysstat con, v$sysstat phy
WHERE cur.name = 'db block gets'
AND con.name = 'consistent gets'
AND phy.name = 'physical reads'
/
```

26. Find Top Disk Reads by User (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT username, sum(physical_reads) AS total_reads
FROM v$session_longops ls
JOIN dba_users u ON ls.user_id = u.user_id
GROUP BY username
ORDER BY total_reads DESC;
```

Script 2:

```
select username users, round(DISK_READS/Executions)
DReadsExec,Executions Exec, DISK_READS DReads,sql_text
from gv$sqlarea a, dba_users b
where a.parsing_user_id = b.user_id
and Executions > 0
and DISK_READS > 100000
order by 2 desc;
```

27. Get OS PID from SID (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT spid FROM v$session WHERE sid = <sid>;
```

Script 2:

```
set lines 123
col USERNAME for a15
col OSUSER for a8
col MACHINE for a15
col PROGRAM for a20
select b.spid, a.username, a.program , a.osuser ,a.machine,
a.sid, a.serial#, a.status from gv$session a, gv$process b
where addr=paddr(+) and sid=&sid;
```

28. Get Active SID of a PL/SQL Object (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT sid FROM v$session WHERE program_name = '<object_name>';
```

Script 2:

```
select sid, sql_id,serial#, status, username, program
from v$session
where PLSQL_ENTRY_OBJECT_ID in (select object_id
from dba_objects
where object_name in ('&PROCEDURE_NAME'));
```

29. Find Buffer Cache Usage (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT * FROM v$db_cache_advice;
```

Script 2:

```
col object_name format a30
col to_total format 999.99
SELECT owner, object_name, object_type, count, (count / value) *
100 to_total
FROM (
SELECT a.owner, a.object_name, a.object_type,
count(*) count
FROM dba_objects a,
x$bh b
WHERE a.object_id = b.obj
and a.owner not in ('SYS', 'SYSTEM')
GROUP BY a.owner, a.object_name, a.object_type
ORDER BY 4),
v$parameter
WHERE name = 'db_cache_size'
AND (count / value) * 100 > .005
ORDER BY to_total desc
/
```

30. Monitor Rollback Transactions (Requires SELECT and Alter ANY TABLE privilege):

Script 1:

```
SELECT * FROM v$rollstat;
```

Script 2:

```
select state,UNDOBLOCKSDONE,UNDOBLOCKSTOTAL,
UNDOBLOCKSDONE/UNDOBLOCKSTOTAL*100
from gv$fast_start_transactions;


alter session set nls_date_format='dd-mon-yyyy hh24:mi:ss';


select usn, state, undoblockstotal "Total", undoblocksdone
"Done", undoblockstotal-undoblocksdone "ToDo",
decode(cputime,0,'unknown',
sysdate+(((undoblockstotal-undoblocksdone) / (undoblocksdone /
cputime)) / 86400)) "Estimated time to complete"
from v$fast_start_transactions;


select a.sid, a.username, b.xidusn, b.used_urec, b.used_ublk
from v$session a, v$transaction b
```

```
where a.saddr=b.ses_addr
order by 5 desc;
```

31. Find Column Usage Statistics (Requires SELECT ANY TABLE privilege):

Script 1:

```
SELECT * FROM user_tab_cols WHERE last_analyzed = NULL;
```

Script 2:

```
set lines 150
set pages 500
col table_name for a20
col column_name for a20
select a.object_name table_name, c.column_name,equality_preds,
equijoin_preds, range_preds, like_preds
from dba_objects a, col_usage$ b, dba_tab_columns c
where a.object_id=b.OBJ#
and c.COLUMN_ID=b.INTCOL#
and a.object_name=c.table_name
and b.obj#=a.object_id
and a.object_name='&table_name'
and a.object_type='TABLE'
and a.owner='&owner'
order by 3 desc,4 desc, 5 desc;
```

32. Get Background Process Details (Requires SELECT ANY TABLE privilege):t

Script 1:

```
SELECT * FROM v$bgstat;
```

Script 2:

```
col ksbddidn for a15
col ksmfsnam for a20
col ksbdddsc for a60
set lines 150 pages 5000
SELECT ksbdd.ksbddidn, ksmfsv.ksmfsnam, ksbdd.ksbdddsc
FROM x$ksbdd ksbdd, x$ksbdp ksbdp, x$ksmfsv ksmfsv
WHERE ksbdd.indx = ksbdp.indx
AND ksbdp.addr = ksmfsv.ksmfsadr
ORDER BY ksbdd.ksbddidn;
```

33. Check if Oracle Database is 32-bit or 64-bit(SELECT Privilege on v$version**):**

Script 1:

```
SELECT * FROM v$version WHERE banner LIKE '%Enterprise Edition%'
OR banner LIKE '%Standard Edition%';
```

Interpretation: 32-bit architecture will have "i386" or "ia64" in the output, while 64-bit will have "x86_64" or "amd64".

Script 2:

```
select
length(addr)*4 || '-bits' word_length
from v$process
where ROWNUM =1;
```

34. Get Oracle License Usage Information(Requires SELECT privilege on dba_user_licenses**):**

Script 1:

```
SELECT owner, table_name, license_name, licenses_used,
max_licenses
FROM dba_user_licenses;
```

Script 2:

```
select
samp.dbid,
fu.name,
samp.version,
detected_usages,
total_samples,
decode(to_char(last_usage_date, 'MM/DD/YYYY, HH:MI:SS'),
NULL, 'FALSE',
to_char(last_sample_date, 'MM/DD/YYYY, HH:MI:SS'), 'TRUE',
'FALSE')
currently_used,
first_usage_date,
last_usage_date,
aux_count,
feature_info,
```

```
last_sample_date,
last_sample_period,
sample_interval,
mt.description
from
wri$_dbu_usage_sample samp,
wri$_dbu_feature_usage fu,
wri$_dbu_feature_metadata mt
where
samp.dbid = fu.dbid and
samp.version = fu.version and
fu.name = mt.name and
fu.name not like '_DBFUS_TEST%' and /* filter out test features
*/
bitand(mt.usg_det_method, 4) != 4 /* filter out disabled
features */;
```

35. Monitor Database Optimizer Processing Rate(Requires SELECT privilege on v$optimizer_statistics**):**

Script 1:

```
SELECT * FROM v$optimizer_statistics WHERE statistic_name IN
('OPTIMIZER_PROCESSING_RATE');
```

Script 2:

```
select OPERATION_NAME, DEFAULT_VALUE from
V$OPTIMIZER_PROCESSING_RATE where OPERATION_NAME
in ('IO_BYTES_PER_SEC','CPU_BYTES_PER_SEC', 'CPU_ROWS_PER_SEC');
```

36. Export Data Pump to ASM Diskgroup:

Script 1:

```
-- Replace with actual values
SET VARIABLE dumpfile =
'/u01/app/oracle/admin/ORCL/dpdump/export.dmp';
SET VARIABLE tablespace_name = 'USERS';
SET VARIABLE diskgroup_name = 'DATA';

-- Use DBMS_METADATA.SET_TABLESPACE_GROUP to set the ASM
```

```
diskgroup
EXECUTE DBMS_METADATA.SET_TABLESPACE_GROUP(tablespace_name,
diskgroup_name);

-- Perform the Data Pump export
EXP DP FULL=Y DIRECTORY=DATA_PUMP DUMPFILE='&dumpfile'
TABLESPACE='&tablespace_name';
```

Privilege: EXECUTE on DBMS_METADATA.SET_TABLESPACE_GROUP and required privileges for Data Pump (e.g., EXP).

Execution:

1. Save the script as `export_to_asm.sql`.
2. Connect to your database.
3. Execute the script by running `@export_to_asm.sql` in your client tool.

Script 2:

```
--Create a directory pointing to asm diskgroup( for dumpfiles)
SQL> create directory SOURCE_DUMP as '+NEWTST/TESTDB2/TEMPFILE';
Directory created
--Create a directory pointing to a normal filesystem ( required
for logfiles)
SQL> create directory EXPLOG as '/export/home/oracle';
Directory created.

--export parfile
dumpfile=test.dmp
logfile=EXPLOG:test.log
directory=SOURCE_DUMP
tables=dbatest.EMPTAB
exclude=statistics
```

37. Explain Plan of SQL_ID from Cursor(Privilege: SELECT on v$sql (Oracle))**:**

Script 1:

```
-- Oracle:
SELECT *
FROM table (DBMS_XPLAN.DISPLAY_CURSOR(sql_id => '<sql_id>'));
```

Script 2:

```
--First get the child number of the sql_id .One sql_id can have
multiple child number( one for each plan_hash_value)
SQL> select sql_id,child_number,plan_hash_value from gv$sql
where sql_id='9n2a2c8pvu6bm';
SQL_ID         CHILD_NUMBER PLAN_HASH_VALUE
-------------  ------------ ---------------
9n2a2c8pvu6bm   1             13761463

--Now get the explain plan for cursor:

SELECT * from
TABLE(DBMS_XPLAN.DISPLAY_CURSOR('&sqlid',&child_number));
```

38. Explain Plan of SQL_ID from AWR (Privilege: SELECT on dba_hist_sqlstat (Oracle))**:**

Script 1:

```
-- Oracle:
SELECT *
FROM table (DBMS_XPLAN.DISPLAY_AWR(sql_id => '<sql_id>',
plan_hash_value => <plan_hash_value>));
```

Script 2:

```
set lines 200
SELECT * FROM table(DBMS_XPLAN.DISPLAY_AWR('&sql_id'));
```

39. Get SQL_Text from SID:

Script 1:

```
-- Oracle:
SELECT sql_text
FROM v$sql
WHERE sid = <sid>;

-- SQL Server:
SELECT text
FROM sys.dm_exec_query_stats
WHERE session_id = <sid>
```

Privilege: SELECT on v$sql (Oracle), VIEW ANY DEFINITION (SQL Server)

Script 2:

```
col sql_text form a80
set lines 120
select sql_text from gv$sqltext where hash_value=
(select sql_hash_value from gv$session where sid=&1 and
inst_id=&inst_id)
order by piece
/
```

40. Explain Plan of a SQL Statement:

Script 1:

```
-- Oracle:
EXPLAIN PLAN FOR SELECT * FROM <table_name>;

-- SQL Server:
SET STATISTICS IO ON;
SELECT * FROM <table_name>;
SET STATISTICS IO OFF;
```

Privilege: EXECUTE on DBMS_SQL package (Oracle)

Script 2:

```
--Generate explain plan
-- Syntax EXPLAIN PLAN FOR < SQL STATEMENT> ;
explain plan for
select count(*) from dbaclass;
--View explain plan
select * from table(dbms_xplan.display);
```

41. Explain Plan of a SQL Baseline:

Script 1:

```
-- Oracle:
SELECT * FROM table
(DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE('<baseline_name>'));
```

Privilege: SELECT on dba_sql_plan_baselines (Oracle)

Script 2:

```
--- SYNTAX
--  SELECT *
FROM   TABLE(DBMS_XPLAN.display_sql_plan_baseline(plan_name=>'<S
QL BASELINE NAME>'));
SELECT *
FROM   TABLE(DBMS_XPLAN.display_sql_plan_baseline(plan_name=>'SQ
L_PLAN_gbhrw1v44209a5b2f7514'));
```

42. Get Bind Values of a SQL_ID:

Script 1:

```
-- Oracle:
SELECT *
FROM v$sql_bind_values
WHERE sql_id = '<sql_id>';
```

Privilege: SELECT on v$sql_bind_values (Oracle)

Script 2:

```
SELECT
sql_id,
b. LAST_CAPTURED,
t.sql_text sql_text,
b.HASH_VALUE,
b.name bind_name,
b.value_string bind_value
FROM
gv$sql t
JOIN
gv$sql_bind_capture b using (sql_id)
WHERE
b.value_string is not null
AND
sql_id='&sqlid'
/
```

43. Flush a SQL Query from Cursor:

Script 1:

```
-- Oracle:
DBMS_SQL.CLOSE_CURSOR(<cursor_id>);
```

Privilege: EXECUTE on DBMS_SQL package (Oracle)

Script 2:

```
--First get the address, hash_value of the sql_id
select ADDRESS, HASH_VALUE from V$SQLAREA where SQL_ID like
'5qd8a442c328k';
ADDRESS          HASH_VALUE
---------------  ------------
C000007067F39FF0  4000666812
--Then flush the query
SQL> exec DBMS_SHARED_POOL.PURGE ('C000007067F39FF0,
4000666812', 'C');
```

Note : For RAC, same need to be executed on all the nodes .

44. Tracing All Sessions of a User:

Script 1:

```
-- Set the trace file name
ALTER SESSION SET TRACE FILE_NAME_PREFIX = '<trace_file_name>';

-- Enable tracing for all sessions of a user
ALTER SESSION SET EVENTS 'trace any ddl' SCOPE=SESSION SID =
'<user_sid>';
```

Note: This requires DBA privileges.

Script 2:

```
-- CREATE THE BELOW TRIGGER TO ENABLE TRACE ALL SESSION OF USER
( SCOTT)
CREATE OR REPLACE TRIGGER USER_TRACE_TRG
AFTER LOGON ON DATABASE
BEGIN
    IF USER = 'SCOTT'
  THEN
    execute immediate 'alter session set events ''10046 trace
name context forever, level 12''';
  END IF;
EXCEPTION
WHEN OTHERS THEN
NULL;
END;
/
```

45. Enable Trace for a Session:

Script 1:

```
-- Set the trace file name
ALTER SESSION SET TRACE FILE_NAME_PREFIX = '<trace_file_name>';

-- Enable tracing for a specific session
ALTER SESSION SET EVENTS 'trace any ddl' SCOPE=SESSION SID =
<sid>;
```

Privilege: DBA privileges

Script 2:

```
EXEC DBMS_SYSTEM.set_sql_trace_in_session(sid=>321,
serial#=>1234, sql_trace=>FALSE);
--Get the trace file name
SELECT p.tracefile FROM v$session s JOIN v$process p ON s.paddr
= p.addr WHERE s.sid = 321;
```

46. Enable 10053 OPTIMIZER TRACE:

Script 1:

```
ALTER SESSION SET EVENTS '10053 trace optimizer' SCOPE=SESSION;
```

Privilege: EXECUTE on DBMS_SQL package

Script 2:

```
Begin
dbms_sqldiag.dump_trace(p_sql_id=>'dmx08r6ayx800',
p_child_number=>0,
p_component=>'Compiler',
p_file_id=>'TEST_OBJ3_TRC');
END;
/
```

47. Enable Trace for a SQL_ID:

```
ALTER SESSION SET EVENTS 'sql_id <sql_id>' SCOPE=SESSION;

ALTER system set events 'sql_trace [sql:8krc88r46raff]';
```

Privilege: DBA privileges

48. Execution Detail of a SQL_ID in the Cursor (SELECT on v$sql, dba_hist_sql_statements):

Script 1:

```
SELECT *
FROM v$sql s
JOIN dba_hist_sql_statements h ON s.sql_id = h.sql_id
WHERE s.sql_id = '<SQL_ID>';
```

Script 2:

```
SELECT
module,parsing_schema_name,inst_id,sql_id,plan_hash_value,child_
number,sql_fulltext,
to_char(last_active_time,'DD/MM/YY HH24:MI:SS'
),sql_plan_baseline,executions,
elapsed_time/executions/1000/1000,rows_processed from gv$sql
where sql_id in ('&sql_id');
```

49. Enable Tracing for a Listener (ALTER SYSTEM, EXECUTE on DBMS_MONITOR):

Script 1:

```
ALTER SYSTEM SET TRACE_FILE_NAME='/path/to/listener_trace.trc';
ALTER SYSTEM SET EVENTS 'listener:listener_event' SCOPE=SYSTEM
SID='*';
DBMS_MONITOR.START_TRACE(TRACE_FILENAME =>
'/path/to/listener_trace.trc', EVENTS =>
'listener:listener_event');
```

Script 2:

```
--Set to the listener you want to trace
LSNRCTL> set cur LISTENER_TEST
-- Enable Trace:
LSNRCTL> set trc_level ADMIN
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER_TEST)))
LISTENER_TEST parameter "trc_level" set to admin
The command completed successfully
```

50. PGA Usage by Sessions (SELECT on v$session):

Script 1:

```
SELECT session_id, username, program, pga_alloc_request_count,
pga_dealloc_request_count
FROM v$session
ORDER BY pga_alloc_request_count DESC;
```

Script 2:

```
set lines 2000
SELECT SID, b.NAME, ROUND(a.VALUE/(1024*1024),2) MB FROM
v$sesstat a, v$statname b
WHERE (NAME LIKE '%session uga memory%' OR NAME LIKE '%session
pga memory%')
AND a.statistic# = b.statistic# order by
ROUND(a.VALUE/(1024*1024),2) desc
```

51. Segments with High Physical Read (SELECT on v$sysstat, dba_segments):

Script 1:

```
SELECT ss.segment_type, s.segment_name, ss.executions,
ss.physical_reads
FROM v$sysstat ss
JOIN dba_segments s ON ss.segment_type = s.segment_type AND
ss.segment_id = s.segment_id
WHERE ss.statistic# = 'physical reads'
ORDER BY ss.physical_reads DESC;
```

Script 2:

```
set pagesize 200
setlinesize 120
col segment_name format a20
col owner format a10
select segment_name,object_type,total_physical_reads
from ( select owner||'.'||object_name as
segment_name,object_type,
value as total_physical_reads
from v$segment_statistics
where statistic_name in ('physical reads')
order by total_physical_reads desc)
where rownum <=10;
```

52. I/O Usage of Each Tempfile (SELECT on v$tempfile):

Script 1:

```
SELECT tempfile_path, total_reads, total_writes,
total_bytes_read, total_bytes_written
FROM v$tempfile;
```

Script 2:

```
SELECT SUBSTR(t.name,1,50) AS file_name,
f.phyblkrd AS blocks_read,
f.phyblkwrt AS blocks_written,
f.phyblkrd + f.phyblkwrt AS total_io
FROM v$tempstat f,v$tempfile t
WHERE t.file# = f.file#
ORDER BY f.phyblkrd + f.phyblkwrt DESC;

select * from (SELECT u.tablespace, s.username, s.sid,
s.serial#, s.logon_time, program, u.extents, ((u.blocks*8)/1024)
as MB,
i.inst_id,i.host_name
FROM gv$session s, gv$sort_usage u ,gv$instance i
WHERE s.saddr=u.session_addr and u.inst_id=i.inst_id order by MB
DESC) a where rownum<10;
```

53. Current SGA Usage (SELECT on v$sgastat):

Script 1:

```
SELECT name, value
FROM v$sgastat
ORDER BY value DESC;
```

Script 2:

```
select round(used.bytes /1024/1024 ,2) used_mb
, round(free.bytes /1024/1024 ,2) free_mb
, round(tot.bytes /1024/1024 ,2) total_mb
from (select sum(bytes) bytes
from v$sgastat
where name != 'free memory') used
, (select sum(bytes) bytes
from v$sgastat
where name = 'free memory') free
, (select sum(bytes) bytes
from v$sgastat) tot
/
```

54. Top Running Queries from ASH (SELECT on dba_hist_active_session_history):

Script 1:

```
SELECT sql_id, elapsed_time_delta, executions, wait_time_delta
FROM dba_hist_active_session_history
ORDER BY elapsed_time_delta DESC
FETCH FIRST 10 ROWS ONLY;
```

Script 2:

```
--Query to get list of top running sqls in PAST between sysdate-
1 to sysdate-23/34 . You can change accordingly
SELECT active_session_history.user_id,
dba_users.username,
sqlarea.sql_text,
SUM(active_session_history.wait_time +
active_session_history.time_waited)/1000000
ttl_wait_time_in_seconds
FROM v$active_session_history active_session_history,
v$sqlarea sqlarea,
dba_users
WHERE active_session_history.sample_time BETWEEN SYSDATE - 1 AND
SYSDATE-23/24
AND active_session_history.sql_id = sqlarea.sql_id
AND active_session_history.user_id = dba_users.user_id
and dba_users.username not in ('SYS','DBSNMP')
GROUP BY active_session_history.user_id,sqlarea.sql_text,
dba_users.username
ORDER BY 4 DESC
/
```

55. Find Blocking Sessions from ASH (SELECT on dba_hist_active_session_history):

Script 1:

```
SELECT blocking_session_status, blocking_session_sql_id,
waiting_session_sql_id
FROM dba_hist_active_session_history
WHERE blocking_session_status = 'ACTIVE'
ORDER BY block_time_delta DESC;
```

Script 2:

```
--Query will list the blocking session details between SYSDATE -
1 AND SYSDATE-23/24 ( PAST)
set pagesize 50
set linesize 120
```

```
col sql_id format a15
col inst_id format '9'
col sql_text format a50
col module format a10
col blocker_ses format '999999'
col blocker_ser format '999999'
SELECT distinct
a.sql_id ,
a.inst_id,
a.blocking_session blocker_ses,
a.blocking_session_serial# blocker_ser,
a.user_id,
s.sql_text,
a.module,a.sample_time
FROM GV$ACTIVE_SESSION_HISTORY a,
gv$sql s
where a.sql_id=s.sql_id
and blocking_session is not null
and a.user_id <> 0 -- exclude SYS user
and a.sample_time BETWEEN SYSDATE - 1 AND SYSDATE-23/24
/
```

56. Top CPU-Consuming Sessions (SELECT on v$session, v$session_longops):

Script 1:

```
SELECT session_id, username, program, cpu_time,
cumulative_cpu_time
FROM v$session s
LEFT JOIN v$session_longops l ON s.sid = l.sid
ORDER BY cpu_time + COALESCE(l.time_waited, 0) DESC;
```

Script 2:

```
col program form a30 heading "Program"
col CPUMins form 99990 heading "CPU in Mins"
select rownum as rank, a.*
from (
SELECT v.sid, program, v.value / (100 * 60) CPUMins
FROM v$statname s , v$sesstat v, v$session sess
WHERE s.name = 'CPU used by this session'
and sess.sid = v.sid
and v.statistic#=s.statistic#
and v.value>0
```

```
ORDER BY v.value DESC) a
where rownum < 11;
```

57. Sessions Holding Library Cache Lock (SELECT on v$resource_lock, dba_objects):

Script 1:

```
SELECT session_id, request, lock_mode, username,
       o.owner, o.object_name
FROM v$resource_lock l
JOIN dba_objects o ON l.resource_name = o.object_id
WHERE type = '2' -- Library Cache
ORDER BY
```

Script 2:

```
--For standalone db:
select sid Waiter, p1raw,
substr(rawtohex(p1),1,30) Handle,
substr(rawtohex(p2),1,30) Pin_addr
from v$session_wait where wait_time=0 and event like '%library
cache%';
--For RAC DB:
select a.sid Waiter,b.SERIAL#,a.event,a.p1raw,
substr(rawtohex(a.p1),1,30) Handle,
substr(rawtohex(a.p2),1,30) Pin_addr
from v$session_wait a,v$session b where a.sid=b.sid
and a.wait_time=0 and a.event like 'library cache%';
--or
set lines 152
col sid for a9999999999999
col name for a40
select a.sid,b.name,a.value,b.class
from gv$sesstat a , gv$statname b
where a.statistic#=b.statistic#
and name like '%library cache%';
```

Objects Locked by Library Cache (SELECT on v$library_cache):

Script 1:

```
SELECT owner, object_name, object_type, lock_mode, session_id
FROM v$library_cache
```

```
WHERE status = 'ACTIVE'
AND lock_mode IN ('H', 'RU');
```

Script 2:

```
select to_char(SESSION_ID,'999') sid ,
substr(LOCK_TYPE,1,30) Type,
substr(lock_id1,1,23) Object_Name,
substr(mode_held,1,4) HELD, substr(mode_requested,1,4) REQ,
lock_id2 Lock_addr
from dba_lock_internal
where
mode_requested'None'
and mode_requestedmode_held
and session_id in ( select sid
from v$session_wait where wait_time=0
and event like '%library cache%') ;
```

58. Sessions Accessing an Object (SELECT on v$session and v$sql):

Script 1:

```
SELECT s.sid, s.serial#, s.username, s.machine, sq.sql_text
FROM v$session s
JOIN v$sql sq ON s.sql_id = sq.sql_id
WHERE sq.sql_text LIKE '%'|| object_name ||'%';
```

Script 2:

```
set lines 299
column object format a30
column owner format a10
select * from gv$access where owner='&OWNER' and
object='&object_name' and
/
```

59. SQLs Doing a Full Table Scan (SELECT on v$sql_plan):

Script 1:

```
SELECT sql_id, operation, object_name, rows
FROM v$sql_plan sp
JOIN v$sql s ON sp.sql_id = s.sql_id
WHERE operation = 'FULL SCAN TABLE'
ORDER BY rows DESC;
```

Script 2:

```
select sql_id,object_owner,object_name from V$SQL_PLAN where
operation='TABLE ACCESS' and
options='FULL' and
object_owner not in ('SYS','SYSTEM','DBSNMP');
```

60. Dictionary Cache Hit Ratio (SELECT on v$object_cache):

Script 1:

```
SELECT 1 - (pin_hit_count / (pin_hit_count + pin_miss_count)) *
100 AS miss_ratio
FROM v$object_cache;
```

Script 2:

```
select sum(gets) as "Gets", sum(getmisses) as "Misses",
(1-(sum(getmisses)/sum(gets)))*100 as "CACHE HIT RATIO"
from gv$rowcache;
```

NOTE - CACHE HIT RATIO SHOULD BE MORE THAN 95 PERCENT.

61. Mutex Sleeps in the Database (SELECT on v$resource_wait_stat):

Script 1:

```
SELECT wait_class, wait_time_waited_micro, time_waited_micro,
event
FROM v$resource_wait_stat
WHERE wait_class = 'Latch'
ORDER BY time_waited_micro DESC;
```

Script 2:

```
column mux format a18 heading 'Mutex Type' trunc;
column loc format a32 heading 'Location' trunc;
column sleeps format 9,999,999,990 heading 'Sleeps';
column wt format 9,999,990.9 heading 'Wait |Time (s)';
```

```
select e.mutex_type mux
, e.location loc
, e.sleeps - nvl(b.sleeps, 0) sleeps
, (e.wait_time - nvl(b.wait_time, 0))/1000000 wt
from DBA_HIST_MUTEX_SLEEP b
, DBA_HIST_MUTEX_SLEEP e
where b.snap_id(+) = &bid
and e.snap_id = &eid
and b.dbid(+) = e.dbid
and b.instance_number(+) = e.instance_number
and b.mutex_type(+) = e.mutex_type
and b.location(+) = e.location
and e.sleeps - nvl(b.sleeps, 0) > 0
order by e.wait_time - nvl(b.wait_time, 0) desc;
```

62. Queries Causing High Physical Read (SELECT on v$sql and v$sql_plan):

Script 1:

```
SELECT s.sql_id, sq.sql_text, sp.operation, sp.rows
FROM v$sql s
JOIN v$sql_plan sp ON sp.sql_id = s.sql_id
JOIN v$session se ON se.sql_id = s.sql_id
WHERE se.status = 'ACTIVE'
AND sp.operation = 'TABLE ACCESS FULL'
AND se.physical_reads > 1000 -- Adjust the threshold as needed
ORDER BY se.physical_reads DESC;
```

Script 2:

```
SELECT schema, sql_text, disk_reads, round(cpu,2) FROM
(SELECT s.parsing_schema_name schema, t.sql_id, t.sql_text,
t.disk_reads,
t.sorts, t.cpu_time/1000000 cpu, t.rows_processed,
t.elapsed_time
FROM v$sqlstats t join v$sql s on(t.sql_id = s.sql_id)
WHERE parsing_schema_name = 'SCOTT'
ORDER BY disk_reads DESC)
WHERE rownum <= 5;
```

63. Objects Causing Latch Contention (SELECT on v$latch and v$latchholder):

Script 1:

```
SELECT l.latch_id, l.name, lh.session_id, lh.sql_id
FROM v$latch l
JOIN v$latchholder lh ON l.latch_id = lh.latch_id
WHERE l.busy_wait_time > 1000 -- Adjust the threshold as needed
ORDER BY l.busy_wait_time DESC;
```

Script 2:

```
col OBJECT_NAME for a30
col owner for a12
with bh_lc as
(select
lc.addr, lc.child#, lc.gets, lc.misses, lc.immediate_gets,
lc.immediate_misses,
lc.spin_gets, lc.sleeps,
bh.hladdr, bh.tch tch, bh.file#, bh.dbablk, bh.class, bh.state,
bh.obj
from
v$session_wait sw,
v$latchname ld,
v$latch_children lc,
x$bh bh
where lc.addr =sw.p1raw
and sw.p2= ld.latch#
and ld.name='cache buffers chains'
and lower(sw.event) like '%latch%'
and bh.hladdr=lc.addr
)
select bh_lc.hladdr, bh_lc.tch, o.owner, o.object_name,
o.object_type,
bh_lc.child#,
bh_lc.gets, bh_lc.misses, bh_lc.immediate_gets,
bh_lc.immediate_misses, spin_gets, sleeps
from
bh_lc, dba_objects o
where bh_lc.obj = o.data_object_id(+)
order by 1,2 desc;
```

64. Latch Type and SQL Hash Value (SELECT on v$latch, v$latchholder):

Script 1:

```
SELECT l.name AS latch_name, lh.sql_id
FROM v$latch l
JOIN v$latchholder lh ON l.latch_id = lh.latch_id
WHERE l.name LIKE '%'|| latch_name ||'%'; -- Replace
'latch_name' with the specific latch you want to investigate
```

Script 2:

```
Set lines 160 pages 100
Column event format A35
Column name format A35
select x.event, x.sql_hash_value,
case when x.event like 'latch%' then
l.name
else ' '
end name,
x.cnt from (
select substr(w.event, 1, 28) event, s.sql_hash_value,
w.p2,count(*) cnt
from v$session_wait w, v$session s, v$process p
where s.sid=w.sid
and p.addr = s.paddr
and s.username is not null
and w.event not like '%pipe%'
and w.event not like 'SQL*%'
group by substr(w.event, 1, 28), sql_hash_value,w.p2
) x,
v$latch l
where
x.p2 = l.latch#(+)
order by cnt;
```

65. Objects Causing Flushing of Shared Pool (SELECT on v$object_cache, v$object_stat):

Script 1:

```
SELECT o.object_name, oc.pin_count, os.dirty_writes
FROM v$object_cache oc
JOIN v$object_stat os ON oc.object_id = os.object_id
JOIN dba_objects o ON o.object_id = oc.object_id
WHERE oc.pin_count > 10
```

Script 2:

```
Set lines 160 pages 100
Select * from x$ksmlru order by ksmlrnum;
```

66. SQL Tuning Advisor for SQL_ID from the Cursor:

Script 1:

```
-- Oracle:
DECLARE
  CURSOR c_sql_stats IS
    SELECT sql_id, sql_text
    FROM v$sql
    WHERE cursor_cache# IS NOT NULL;
  l_sql_id VARCHAR2(30);
  l_sql_text VARCHAR2(4000);
BEGIN
  OPEN c_sql_stats;
  LOOP
    FETCH c_sql_stats INTO l_sql_id, l_sql_text;
    EXIT WHEN c_sql_stats%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Analyzing SQL_ID: ' || l_sql_id);

    -- Call the SQL Tuning Advisor
    DBMS_SQLTUNE.ADVISE(
      sql_id => l_sql_id,
      tuning_set => 'OPTIMIZATION_SET',  -- Adjust set as needed
      tuning_task_name => 'tuning_' || l_sql_id
    );
  END LOOP;
  CLOSE c_sql_stats;
END;
/
```

Privilege: EXECUTE on DBMS_SQLTUNE package

Script 2:

```
--Create tuning task
set long 1000000000
DECLARE
l_sql_tune_task_id VARCHAR2(100);
BEGIN
l_sql_tune_task_id := DBMS_SQLTUNE.create_tuning_task (
sql_id => 'apwfwjhgc9sk8',
scope => DBMS_SQLTUNE.scope_comprehensive,
time_limit => 500,
task_name => 'apwfwjhgc9sk8_tuning_task_1',
description => 'Tuning task for statement apwfwjhgc9sk8');
DBMS_OUTPUT.put_line('l_sql_tune_task_id: ' ||
l_sql_tune_task_id);
END;
/
--Execute tuning task
EXEC DBMS_SQLTUNE.execute_tuning_task(task_name =>
'apwfwjhgc9sk8_tuning_task_1');
--Generate report
SET LONG 10000000;
SET PAGESIZE 100000000
SET LINESIZE 200
SELECT
DBMS_SQLTUNE.report_tuning_task('apwfwjhgc9sk8_tuning_task_1')
AS recommendations FROM dual;
SET PAGESIZE 24
```

67. Run SGA Target Advisory:

Script 1:

```
-- Oracle:
BEGIN
  DBMS_RESOURCE_MANAGER.RUN_TARGET_ADVISORY(
    target_type => 'SGA_TARGET',
    advisor_name => 'SGA_TARGET_ADVISOR',
    advisor_task_name => 'SGA_TARGET_ADVISORY'
  );
END;
/
```

Privilege: EXECUTE on DBMS_RESOURCE_MANAGER package

Script 2:

```
--STATISTICS_LEVEL should be TYPICAL/ALL.
SQL> show parameter statistics_level
NAME TYPE VALUE
---------------------------------- ---------------------------
----- --------------------------
statistics_level string TYPICAL
select * from v$sga_target_advice order by sga_size;
```

68. Run Shared Pool Advisory:

Script 1:

```
-- Oracle:
BEGIN
  DBMS_RESOURCE_MANAGER.RUN_TARGET_ADVISORY(
    target_type => 'SHARED_POOL_TARGET',
    advisor_name => 'SHARED_POOL_ADVISOR',
    advisor_task_name => 'SHARED_POOL_ADVISORY'
  );
END;
/
```

Privilege: EXECUTE on DBMS_RESOURCE_MANAGER package

Script 2:

```
SELECT shared_pool_size_for_estimate "Size of Shared Pool in
MB",
shared_pool_size_factor "Size Factor",
estd_lc_time_saved "Time Saved in sec" FROM
v$shared_pool_advice;
```

69. Generate AWR Report:

Script 1:

```
-- Oracle:
DBMS_SNAPSHOT.CREATE_SNAPSHOT(<snap_id>, AS OF SYSDATE -
<retention_days>);
```

Privilege: CREATE ANY SNAPSHOT and SELECT on DBA_SNAPSHOTS

Script 2:

70. Generate ADDM Report:

```
-- Oracle:
DBMS_AWRM.RUN_REPORT(
  report_type => DBMS_AWRM.REPORT_TYPE_ADDM,
  snap_id => <snap_id>, -- Replace with the snapshot ID
  instance_number => DBMS_AWRM.INSTANCE_NUMBER,
  output_file => 'addm_report.sql'
);
```

Privilege: EXECUTE on DBMS_AWRM package

```
cd $ORACLE_HOME/rdbms/admin
SQL> @addm_report.sql
Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 1058
Begin Snapshot Id specified: 1058
Enter value for end_snap: 1059
End Snapshot Id specified: 1059
```

Remember:

- Adjust the scripts and privileges based on your specific database platform.
- Be cautious granting EXECUTE privileges on packages like DBMS_SQLTUNE and DBMS_RESOURCE_MANAGER.
- Consider performance implications before running resource-intensive queries on production systems.

Additional Tips:

- Consider using tools and utilities provided by Oracle for comprehensive monitoring and performance analysis.
- Regularly review and adapt your monitoring strategy based on your specific needs and performance goals.
- Optimize your queries for efficiency by utilizing appropriate indexes, views, and filtering techniques.

By following these guidelines and using the provided queries and scripts, you can efficiently monitor various aspects of your Oracle database environment. Remember to adapt the information and commands based on your specific context and database configuration.