**Find Long Running Sessions-**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Check the ALL Active/Inactive session

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

set linesize 750 pages 9999

column box format a30

column spid format a10

column username format a30

column program format a30

column os\_user format a20

col LOGON\_TIME for a20

select b.inst\_id,b.sid,b.serial#,a.spid, substr(b.machine,1,30) box,to\_char (b.logon\_time, 'dd-mon-yyyy hh24:mi:ss') logon\_time,

substr(b.username,1,30) username,

substr(b.osuser,1,20) os\_user,

substr(b.program,1,30) program,status,b.last\_call\_et AS last\_call\_et\_secs,b.sql\_id

from gv$session b,gv$process a

where b.paddr = a.addr

and a.inst\_id = b.inst\_id

and type='USER'

order by logon\_time;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Check the all Active session

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

set linesize 750 pages 9999

column box format a30

column spid format a10

column username format a30

column program format a30

column os\_user format a20

col LOGON\_TIME for a20

select b.inst\_id,b.sid,b.serial#,a.spid, substr(b.machine,1,30) box,to\_char (b.logon\_time, 'dd-mon-yyyy hh24:mi:ss') logon\_time,

substr(b.username,1,30) username,

substr(b.osuser,1,20) os\_user,

substr(b.program,1,30) program,status,b.last\_call\_et AS last\_call\_et\_secs,b.sql\_id

from gv$session b,gv$process a

where b.paddr = a.addr

and a.inst\_id = b.inst\_id

and type='USER' and b.status='ACTIVE'

order by logon\_time;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Check the ALL Active/Inactive sessions by SID

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

set linesize 750 pages 9999

column box format a30

column spid format a10

column username format a30

column program format a30

column os\_user format a20

col LOGON\_TIME for a20

select b.inst\_id,b.sid,b.serial#,a.spid, substr(b.machine,1,30) box,to\_char (b.logon\_time, 'dd-mon-yyyy hh24:mi:ss') logon\_time,

substr(b.username,1,30) username,

substr(b.osuser,1,20) os\_user,

substr(b.program,1,30) program,status,b.last\_call\_et AS last\_call\_et\_secs,b.sql\_id

from gv$session b,gv$process a

where b.paddr = a.addr

and a.inst\_id = b.inst\_id

and type='USER' and b.SID=**'&SID'**

-- and b.status='ACTIVE'

-- and b.status='INACTIVE'

order by logon\_time;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Check the ALL Active/Inactive sessions by Username

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

set linesize 750 pages 9999

column box format a30

column spid format a10

column username format a30

column program format a30

column os\_user format a20

col LOGON\_TIME for a20

select b.inst\_id,b.sid,b.serial#,a.spid, substr(b.machine,1,30) box,to\_char (b.logon\_time, 'dd-mon-yyyy hh24:mi:ss') logon\_time,

substr(b.username,1,30) username,

substr(b.osuser,1,20) os\_user,

substr(b.program,1,30) program,status,b.last\_call\_et AS last\_call\_et\_secs,b.sql\_id

from gv$session b,gv$process a

where b.paddr = a.addr

and a.inst\_id = b.inst\_id

and type='USER' and b.username**='&username'**

-- and b.status='ACTIVE'

-- and b.status='INACTIVE'

order by logon\_time;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL Monitor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

set lines 1000 pages 9999

column sid format 9999

column serial for 999999

column status format a15

column username format a10

column sql\_text format a80

column module format a30

col program for a30

col SQL\_EXEC\_START for a20

SELECT \* FROM

(SELECT status,inst\_id,sid,SESSION\_SERIAL# as Serial,username,sql\_id,SQL\_PLAN\_HASH\_VALUE,

MODULE,program,

TO\_CHAR(sql\_exec\_start,'dd-mon-yyyy hh24:mi:ss') AS sql\_exec\_start,

ROUND(elapsed\_time/1000000) AS "Elapsed (s)",

ROUND(cpu\_time /1000000) AS "CPU (s)",

substr(sql\_text,1,30) sql\_text

FROM gv$sql\_monitor where status='EXECUTING' and module not like '%emagent%'

ORDER BY sql\_exec\_start desc

);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

---- Sql-Monitor report for a sql\_id ( Like OEM report)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

column text\_line format a254

set lines 750 pages 9999

set long 20000 longchunksize 20000

select

dbms\_sqltune.report\_sql\_monitor\_list() text\_line

from dual;

select

dbms\_sqltune.report\_sql\_monitor() text\_line

from dual;

**4. Blocking sessions**

\*\*\*\* To find Blocking GOOD query

set lines 750 pages 9999

col blocking\_status for a100

select s1.inst\_id,s2.inst\_id,s1.username || '@' || s1.machine

|| ' ( SID=' || s1.sid || ' ) is blocking '

|| s2.username || '@' || s2.machine || ' ( SID=' || s2.sid || ' ) ' AS blocking\_status

from gv$lock l1, gv$session s1, gv$lock l2, gv$session s2

where s1.sid=l1.sid and s2.sid=l2.sid and s1.inst\_id=l1.inst\_id and s2.inst\_id=l2.inst\_id

and l1.BLOCK=1 and l2.request > 0

and l1.id1 = l2.id1

and l2.id2 = l2.id2

order by s1.inst\_id;

\*\*\*\* Check who is blocking who in RAC, including objects

SELECT DECODE(request,0,'Holder: ','Waiter: ') || gv$lock.sid sess, machine, do.object\_name as locked\_object,id1, id2, lmode, request, gv$lock.type

FROM gv$lock join gv$session on gv$lock.sid=gv$session.sid and gv$lock.inst\_id=gv$session.inst\_id

join gv$locked\_object lo on gv$lock.SID = lo.SESSION\_ID and gv$lock.inst\_id=lo.inst\_id

join dba\_objects do on lo.OBJECT\_ID = do.OBJECT\_ID

WHERE (id1, id2, gv$lock.type) IN (

SELECT id1, id2, type FROM gv$lock WHERE request>0)

ORDER BY id1, request;

**5. Kill Sessions**

select 'alter system kill session ' || '''' || sid || ',' || serial# ||',@'|| inst\_id || '''' || ' immediate;' from gv$session where sid='&sid';

**6. SQL History**

set lines 1000 pages 9999

COL instance\_number FOR 9999 HEA 'Inst';

COL end\_time HEA 'End Time';

COL plan\_hash\_value HEA 'Plan|Hash Value';

COL executions\_total FOR 999,999 HEA 'Execs|Total';

COL rows\_per\_exec HEA 'Rows Per Exec';

COL et\_secs\_per\_exec HEA 'Elap Secs|Per Exec';

COL cpu\_secs\_per\_exec HEA 'CPU Secs|Per Exec';

COL io\_secs\_per\_exec HEA 'IO Secs|Per Exec';

COL cl\_secs\_per\_exec HEA 'Clus Secs|Per Exec';

COL ap\_secs\_per\_exec HEA 'App Secs|Per Exec';

COL cc\_secs\_per\_exec HEA 'Conc Secs|Per Exec';

COL pl\_secs\_per\_exec HEA 'PLSQL Secs|Per Exec';

COL ja\_secs\_per\_exec HEA 'Java Secs|Per Exec';

SELECT 'gv$dba\_hist\_sqlstat' source,h.instance\_number,

TO\_CHAR(CAST(s.begin\_interval\_time AS DATE), 'DD-MM-YYYY HH24:MI') snap\_time,

TO\_CHAR(CAST(s.end\_interval\_time AS DATE), 'DD-MM-YYYY HH24:MI') end\_time,

h.sql\_id,

h.plan\_hash\_value,

h.executions\_total,

TO\_CHAR(ROUND(h.rows\_processed\_total / h.executions\_total), '999,999,999,999') rows\_per\_exec,

TO\_CHAR(ROUND(h.elapsed\_time\_total / h.executions\_total / 1e6, 3), '999,990.000') et\_secs\_per\_exec,

TO\_CHAR(ROUND(h.cpu\_time\_total / h.executions\_total / 1e6, 3), '999,990.000') cpu\_secs\_per\_exec,

TO\_CHAR(ROUND(h.iowait\_total / h.executions\_total / 1e6, 3), '999,990.000') io\_secs\_per\_exec,

TO\_CHAR(ROUND(h.clwait\_total / h.executions\_total / 1e6, 3), '999,990.000') cl\_secs\_per\_exec,

TO\_CHAR(ROUND(h.apwait\_total / h.executions\_total / 1e6, 3), '999,990.000') ap\_secs\_per\_exec,

TO\_CHAR(ROUND(h.ccwait\_total / h.executions\_total / 1e6, 3), '999,990.000') cc\_secs\_per\_exec,

TO\_CHAR(ROUND(h.plsexec\_time\_total / h.executions\_total / 1e6, 3), '999,990.000') pl\_secs\_per\_exec,

TO\_CHAR(ROUND(h.javexec\_time\_total / h.executions\_total / 1e6, 3), '999,990.000') ja\_secs\_per\_exec

FROM dba\_hist\_sqlstat h,

dba\_hist\_snapshot s

WHERE h.sql\_id = **'&sql\_id'**

AND h.executions\_total > 0

AND s.snap\_id = h.snap\_id

AND s.dbid = h.dbid

AND s.instance\_number = h.instance\_number

UNION ALL

SELECT 'gv$sqlarea\_plan\_hash' source,h.inst\_id,

TO\_CHAR(sysdate, 'DD-MM-YYYY HH24:MI') snap\_time,

TO\_CHAR(sysdate, 'DD-MM-YYYY HH24:MI') end\_time,

h.sql\_id,

h.plan\_hash\_value,

h.executions,

TO\_CHAR(ROUND(h.rows\_processed / h.executions), '999,999,999,999') rows\_per\_exec,

TO\_CHAR(ROUND(h.elapsed\_time / h.executions / 1e6, 3), '999,990.000') et\_secs\_per\_exec,

TO\_CHAR(ROUND(h.cpu\_time / h.executions / 1e6, 3), '999,990.000') cpu\_secs\_per\_exec,

TO\_CHAR(ROUND(h.USER\_IO\_WAIT\_TIME / h.executions / 1e6, 3), '999,990.000') io\_secs\_per\_exec,

TO\_CHAR(ROUND(h.CLUSTER\_WAIT\_TIME / h.executions / 1e6, 3), '999,990.000') cl\_secs\_per\_exec,

TO\_CHAR(ROUND(h.APPLICATION\_WAIT\_TIME / h.executions / 1e6, 3), '999,990.000') ap\_secs\_per\_exec,

TO\_CHAR(ROUND(h.CLUSTER\_WAIT\_TIME / h.executions / 1e6, 3), '999,990.000') cc\_secs\_per\_exec,

TO\_CHAR(ROUND(h.PLSQL\_EXEC\_TIME / h.executions / 1e6, 3), '999,990.000') pl\_secs\_per\_exec,

TO\_CHAR(ROUND(h.JAVA\_EXEC\_TIME / h.executions / 1e6, 3), '999,990.000') ja\_secs\_per\_exec

FROM gv$sqlarea\_plan\_hash h

WHERE h.sql\_id = **'&sql\_id'**

AND h.executions > 0

order by source ;

**7. Find Force Matching Signature**

col exact\_matching\_signature for 99999999999999999999999999

col sql\_text for a50

set long 20000

set lines 750 pages 9999

select sql\_id, exact\_matching\_signature, force\_matching\_signature, SQL\_TEXT from v$sqlarea where sql\_id='&sql\_id';

UNION ALL

select sql\_id,force\_matching\_signature,SQL\_TEXT from dba\_hist\_sqltext where sql\_id=**'&sql\_id'**

/

**-- If you want to find Bind variable for SQL\_ID**

col VALUE\_STRING for a50

SELECT NAME,POSITION,DATATYPE\_STRING,VALUE\_STRING FROM gv$sql\_bind\_capture WHERE sql\_id=**'&sql\_id';**

**8. SQL Tuning Advisor**

[**http://www.br8dba.com/sql-tuning-advisor-manually/**](http://www.br8dba.com/sql-tuning-advisor-manually/)

**9. SQLT**

**[http://www.br8dba.com/sqlt/](http://www.br8dba.com/sqlt/" \t "_blank)**

**10. SQL Health Check**

**SQL Tuning Health-Check Script (SQLHC) (Doc ID 1366133.1)**

**11. SQL Plan Flip**

set lines 1000 pages 9999

COL instance\_number FOR 9999 HEA 'Inst';

COL end\_time HEA 'End Time';

COL plan\_hash\_value HEA 'Plan|Hash Value';

COL executions\_total FOR 999,999 HEA 'Execs|Total';

COL rows\_per\_exec HEA 'Rows Per Exec';

COL et\_secs\_per\_exec HEA 'Elap Secs|Per Exec';

COL cpu\_secs\_per\_exec HEA 'CPU Secs|Per Exec';

COL io\_secs\_per\_exec HEA 'IO Secs|Per Exec';

COL cl\_secs\_per\_exec HEA 'Clus Secs|Per Exec';

COL ap\_secs\_per\_exec HEA 'App Secs|Per Exec';

COL cc\_secs\_per\_exec HEA 'Conc Secs|Per Exec';

COL pl\_secs\_per\_exec HEA 'PLSQL Secs|Per Exec';

COL ja\_secs\_per\_exec HEA 'Java Secs|Per Exec';

SELECT 'gv$dba\_hist\_sqlstat' source,h.instance\_number,

TO\_CHAR(CAST(s.begin\_interval\_time AS DATE), 'DD-MM-YYYY HH24:MI') snap\_time,

TO\_CHAR(CAST(s.end\_interval\_time AS DATE), 'DD-MM-YYYY HH24:MI') end\_time,

h.sql\_id,

h.plan\_hash\_value,

h.executions\_total,

TO\_CHAR(ROUND(h.rows\_processed\_total / h.executions\_total), '999,999,999,999') rows\_per\_exec,

TO\_CHAR(ROUND(h.elapsed\_time\_total / h.executions\_total / 1e6, 3), '999,990.000') et\_secs\_per\_exec,

TO\_CHAR(ROUND(h.cpu\_time\_total / h.executions\_total / 1e6, 3), '999,990.000') cpu\_secs\_per\_exec,

TO\_CHAR(ROUND(h.iowait\_total / h.executions\_total / 1e6, 3), '999,990.000') io\_secs\_per\_exec,

TO\_CHAR(ROUND(h.clwait\_total / h.executions\_total / 1e6, 3), '999,990.000') cl\_secs\_per\_exec,

TO\_CHAR(ROUND(h.apwait\_total / h.executions\_total / 1e6, 3), '999,990.000') ap\_secs\_per\_exec,

TO\_CHAR(ROUND(h.ccwait\_total / h.executions\_total / 1e6, 3), '999,990.000') cc\_secs\_per\_exec,

TO\_CHAR(ROUND(h.plsexec\_time\_total / h.executions\_total / 1e6, 3), '999,990.000') pl\_secs\_per\_exec,

TO\_CHAR(ROUND(h.javexec\_time\_total / h.executions\_total / 1e6, 3), '999,990.000') ja\_secs\_per\_exec

FROM dba\_hist\_sqlstat h,

dba\_hist\_snapshot s

WHERE h.sql\_id = **'&sql\_id'**

AND h.executions\_total > 0

AND s.snap\_id = h.snap\_id

AND s.dbid = h.dbid

AND s.instance\_number = h.instance\_number

UNION ALL

SELECT 'gv$sqlarea\_plan\_hash' source,h.inst\_id,

TO\_CHAR(sysdate, 'DD-MM-YYYY HH24:MI') snap\_time,

TO\_CHAR(sysdate, 'DD-MM-YYYY HH24:MI') end\_time,

h.sql\_id,

h.plan\_hash\_value,

h.executions,

TO\_CHAR(ROUND(h.rows\_processed / h.executions), '999,999,999,999') rows\_per\_exec,

TO\_CHAR(ROUND(h.elapsed\_time / h.executions / 1e6, 3), '999,990.000') et\_secs\_per\_exec,

TO\_CHAR(ROUND(h.cpu\_time / h.executions / 1e6, 3), '999,990.000') cpu\_secs\_per\_exec,

TO\_CHAR(ROUND(h.USER\_IO\_WAIT\_TIME / h.executions / 1e6, 3), '999,990.000') io\_secs\_per\_exec,

TO\_CHAR(ROUND(h.CLUSTER\_WAIT\_TIME / h.executions / 1e6, 3), '999,990.000') cl\_secs\_per\_exec,

TO\_CHAR(ROUND(h.APPLICATION\_WAIT\_TIME / h.executions / 1e6, 3), '999,990.000') ap\_secs\_per\_exec,

TO\_CHAR(ROUND(h.CLUSTER\_WAIT\_TIME / h.executions / 1e6, 3), '999,990.000') cc\_secs\_per\_exec,

TO\_CHAR(ROUND(h.PLSQL\_EXEC\_TIME / h.executions / 1e6, 3), '999,990.000') pl\_secs\_per\_exec,

TO\_CHAR(ROUND(h.JAVA\_EXEC\_TIME / h.executions / 1e6, 3), '999,990.000') ja\_secs\_per\_exec

FROM gv$sqlarea\_plan\_hash h

WHERE h.sql\_id = **'&sql\_id'**

AND h.executions > 0

order by source ;

**12. Find Stale Stats**

[**http://www.br8dba.com/statistics/**](http://www.br8dba.com/statistics/)

exec DBMS\_STATS.FLUSH\_DATABASE\_MONITORING\_INFO;

select OWNER,TABLE\_NAME,LAST\_ANALYZED,STALE\_STATS from DBA\_TAB\_STATISTICS where STALE\_STATS='YES' and OWNER=**'&owner;**

**\*\*\* statistics of objects of a specific sql id**

set lines 300 set pages 300

col table\_name for a40

col owner for a30

select distinct owner, table\_name, STALE\_STATS, last\_analyzed, stattype\_locked

from dba\_tab\_statistics

where (owner, table\_name) in

(select distinct owner, table\_name

from dba\_tables

where ( table\_name)

in ( select object\_name

from gv$sql\_plan

where upper(sql\_id) = upper('&sql\_id') and object\_name is not null))

--and STALE\_STATS='YES'

/

**13. Gather Stats**

<http://www.br8dba.com/statistics/>

<http://www.br8dba.com/oracle-histograms/>

**14. PIN Optimal Plan**

**-- Run below script, you can download this script from SQLT**

-- Reference: [**http://www.br8dba.com/how-to-create-custom-sql-profile/**](http://www.br8dba.com/how-to-create-custom-sql-profile/)

**@coe\_xfr\_sql\_monitor.sql <sql\_id> <plan\_hash\_value>** -- to create SQL profile

The coe\_xfr\_sql\_profile.sql script would create another sql file, which should be run to create manual sql profile for the sql

The new sql file created.

ls -ltr coe\_xfr\_sql\_profile\_**<sql\_id>\_<plan\_hash\_value>**.sql

cat coe\_xfr\_sql\_profile\_**<sql\_id>\_<plan\_hash\_value>**.sql

@coe\_xfr\_sql\_profile\_**<sql\_id>\_<plan\_hash\_value>**.sql

**15. Find Fragmentation**

**\*\*\* Table Fragmentation**

select

table\_name,round((blocks\*8),2) "size (kb)" ,

round((num\_rows\*avg\_row\_len/1024),2) "actual\_data (kb)",

(round((blocks\*8),2) - round((num\_rows\*avg\_row\_len/1024),2)) "wasted\_space (kb)"

from

dba\_tables

where owner=**'&OWNER'** and table\_name='**&TABLE\_NAME**' and

(round((blocks\*8),2) > round((num\_rows\*avg\_row\_len/1024),2))

order by 4 desc;

**16. De-Fragmentation**

**\*\*\* There are many methods.**

**Option 1: Shrink command**

alter table enable row movement;

/\*

Using the enable row movement clause can reduce the clustering\_factor for a primary access index, causing excessive I/O. Oracle introduced the sorted gash cluster as a way to keep an index in-order with the table rows, a technique that greatly reduces I/O for common queries.

Beware that using "enable row movement" can cause Oracle to move rows to discontinuous data blocks, causing a performance problem. Remember, the physical placement of rows on data blocks can still make a huge difference in query performance.

\*/

alter table shrink space compact;

alter table shrink space cascade;

http://www.dba-oracle.com/t\_enable\_row\_movement.htm

**Option 2: Table move**

Alter table move - The alter table xxx move command moves rows down into un-used space and adjusts the HWM but does not adjust the segments extents, and the table size remains the same. The alter table move syntax also preserves the index and constraint definitions.

**ALTER TABLE <table\_name> MOVE;**

**Option 3: EXPORT / IMPORT**

\*\* EXPORT

\*\* DROP ALL RESPECTIVE OBJECTS

\*\* IMPORT FROM EXPORT BACKUP

**Option 4: EXPORT / IMPORT WITH TABLE\_EXISTS\_ACTION=REPLACE**

\*\* EXPORT

\*\* IMPORT USING TABLE\_EXISTS\_ACTION=REPLACE

**Option 5: Other methods**