At times, this query is executed in 20+ sessions, lasting hours and hours:

```
SELECT PROCESSINSTANCEID, PROCESSID, START_DATE, END_DATE, STATUS, PARENTPROCESSINSTANCEID, OUTCOME, DURATION,
 USER_IDENTITY, PROCESSVERSION, PROCESSNAME, CORRELATIONKEY, EXTERNALID, PROCESSINSTANCEDESCRIPTION, SLA_DUE_DATE,
SLACOMPLIANCE, LASTMODIFICATIONDATE, ERRORCOUNT
 FROM
 ( SELECT LOG.PROCESSINSTANCEID, LOG.PROCESSID, LOG.START_DATE, LOG.END_DATE, LOG.STATUS, LOG.PARENTPROCESSINSTANCEID,   LOG.OUTCOME,
LOG.DURATION, LOG.USER_IDENTITY, LOG.PROCESSVERSION, LOG.PROCESSNAME, LOG.CORRELATIONKEY, LOG.EXTERNALID,
LOG.PROCESSINSTANCEDESCRIPTION, LOG.SLA_DUE_DATE, LOG.SLACOMPLIANCE, COALESCE ( INFO.LASTMODIFICATIONDATE,   LOG.END_DATE ) AS
LASTMODIFICATIONDATE,
 (
 SELECT COUNT ( ERRINFO.ID )
 FROM EXECUTIONERRORINFO ERRINFO
 WHERE ERRINFO.PROCESS_INST_ID=LOG.PROCESSINSTANCEID
 AND ERRINFO.ERROR_ACK=0
 ) AS ERRORCOUNT
 FROM PROCESSINSTANCELOG LOG
 LEFT JOIN PROCESSINSTANCEINFO INFO ON INFO.INSTANCEID=LOG.PROCESSINSTANCEID
 ) "dbSQL"
 WHERE ERRORCOUNT > 0.0
 ORDER BY START_DATE DESC FETCH FIRST 10 ROWS ONLY
```

The tables:

- PROCESSINSTANCELOG has 56M rows
- PROCESSINSTANCEINFO has 250K rows
- EXECUTIONERRORINFO has 3K rows

This is what happens:

- The table PROCESSINSTANCELOG is full table scanned and PROCESSINSTANCEINFO is outer joined. This alone already takes something like a a minute
- However, the COUNT subquery is where things go awry. This subquery has to be executed for each and every row that is returned from PROCESSINSTANCELOG. In other terms: we run 56M counts in a table of 3K rows! This takes more than an hour.
- What follows only just takes a few seconds, but for completeness sake:
    - o the 56M rows, completed with the ERRORCOUNT, are scanned and only the ones with an ERRORCOUNT > 0 are withheld.
    - o The remaining rows are sorted and only the first 10 rows are returned

So, we do an awful lot of work to only just return ten rows.

How can we avoid the COUNT-subquery? Two possibilities:

1. Do not COUNT the rows “inline”, but use an OUTER JOIN instead. Modifications are in red

```
SELECT PROCESSINSTANCEID, PROCESSID, START_DATE, END_DATE, STATUS, PARENTPROCESSINSTANCEID, OUTCOME, DURATION,
USER_IDENTITY, PROCESSVERSION, PROCESSNAME, CORRELATIONKEY, EXTERNALID, PROCESSINSTANCEDESCRIPTION, SLA_DUE_DATE,
SLACOMPLIANCE, LASTMODIFICATIONDATE, ERRORCOUNT
 FROM
 ( SELECT LOG.PROCESSINSTANCEID, LOG.PROCESSID, LOG.START_DATE, LOG.END_DATE, LOG.STATUS, LOG.PARENTPROCESSINSTANCEID,   LOG.OUTCOME,
LOG.DURATION, LOG.USER_IDENTITY, LOG.PROCESSVERSION, LOG.PROCESSNAME, LOG.CORRELATIONKEY, LOG.EXTERNALID,
LOG.PROCESSINSTANCEDESCRIPTION, LOG.SLA_DUE_DATE, LOG.SLACOMPLIANCE, COALESCE ( INFO.LASTMODIFICATIONDATE,   LOG.END_DATE ) AS
LASTMODIFICATIONDATE,
 -- the following line ...
 eeinfo.errorcount AS ERRORCOUNT
 -- ... replaces this COUNT subquery (commented out)
 --- ( SELECT COUNT ( ERRINFO.ID )
 --- FROM EXECUTIONERRORINFO ERRINFO
 --- WHERE ERRINFO.PROCESS_INST_ID=LOG.PROCESSINSTANCEID
 --- AND ERRINFO.ERROR_ACK=0 ) AS ERRORCOUNT
 FROM PROCESSINSTANCELOG LOG
 --
 LEFT JOIN PROCESSINSTANCEINFO INFO
 ON INFO.INSTANCEID=LOG.PROCESSINSTANCEID
 --
 LEFT OUTER JOIN (
 select PROCESS_INST_ID, count(*) as ERRORCOUNT
 from EXECUTIONERRORINFO
 where ERROR_ACK=0
 group by PROCESS_INST_ID
 having count(*) > 0
 ) eeinfo
 on eeinfo.PROCESS_INST_ID = LOG.PROCESSINSTANCEID
 --
 ) "dbSQL"
 WHERE ERRORCOUNT > 0.0
 ORDER BY START_DATE DESC FETCH FIRST 10 ROWS ONLY
/
```

Response time? The query returns even before my finger leaves the ENTER button:

```
11:58:21 PENEXIS> SELECT PROCESSINSTANCEID, PROCESSID, START_DATE, END_DATE, STATUS, PARENTPROCESSINSTANCEID, OUTCOME, DURATION,
11:58:21   2          USER_IDENTITY, PROCESSVERSION, PROCESSNAME, CORRELATIONKEY, EXTERNALID, PROCESSINSTANCEDESCRIPTION, SLA_DUE_DATE,
11:58:21   3          SLACOMPLIANCE, LASTMODIFICATIONDATE, ERRORCOUNT
11:58:21   4     FROM
11:58:21   5          ( SELECT LOG.PROCESSINSTANCEID, LOG.PROCESSID, LOG.START_DATE, LOG.END_DATE, LOG.STATUS, LOG.PARENTPROCESSINSTANCEID,
11:58:21   6                   LOG.OUTCOME, LOG.DURATION, LOG.USER_IDENTITY, LOG.PROCESSVERSION, LOG.PROCESSNAME, LOG.CORRELATIONKEY, LOG.EXTERNALID,
11:58:21   7                   LOG.PROCESSINSTANCEDESCRIPTION, LOG.SLA_DUE_DATE, LOG.SLACOMPLIANCE, COALESCE ( INFO.LASTMODIFICATIONDATE,
11:58:21   8                   LOG.END_DATE ) AS LASTMODIFICATIONDATE,
11:58:21   9                   eeinfo.errorcount AS ERRORCOUNT
11:58:21  10                ---  ( SELECT COUNT ( ERRINFO.ID )
11:58:21  11                ---      FROM EXECUTIONERRORINFO ERRINFO
11:58:21  12                ---     WHERE ERRINFO.PROCESS_INST_ID=LOG.PROCESSINSTANCEID
11:58:21  13                ---       AND ERRINFO.ERROR_ACK=0 ) AS ERRORCOUNT
11:58:21  14             FROM PROCESSINSTANCELOG LOG
11:58:21  15               --
11:58:21  16                  LEFT JOIN PROCESSINSTANCEINFO INFO
11:58:21  17                  ON INFO.INSTANCEID=LOG.PROCESSINSTANCEID
11:58:21  18               --
11:58:21  19                  LEFT OUTER JOIN (
11:58:21  20                                   select PROCESS_INST_ID, count(*) as ERRORCOUNT
11:58:21  21                                     from EXECUTIONERRORINFO
11:58:21  22                                    where ERROR_ACK=0
11:58:21  23                                    group by PROCESS_INST_ID
11:58:21  24                                    having count(*) > 0
11:58:21  25                                  ) eeinfo
11:58:21  26                   on eeinfo.PROCESS_INST_ID = LOG.PROCESSINSTANCEID
11:58:21  27             --
11:58:21  28          ) "dbSQL"
11:58:21  29    WHERE ERRORCOUNT > 0.0
11:58:21  30    ORDER BY START_DATE DESC FETCH FIRST 10 ROWS ONLY
11:58:21  31  /

PROCESSINSTANCEID PROCESSID                                                        START_DATE                        END_DATE                          STATUS P
----------------- ------------------------------------------------------------- --------------------------------- --------------------------------- ------ -
         97227787 workflows.WaitComTaskToCompleteWorkflow                       03-MAY-22 04.40.27.307000 PM      03-MAY-22 07.26.58.358000 PM           2
         97209624 workflows.CheckSendAdministrativeStatusUsagepoint             03-MAY-22 09.53.15.001000 AM      04-MAY-22 04.49.22.434000 PM           3
         97209567 workflows.CheckSendAdministrativeStatusUsagepoint             03-MAY-22 09.52.24.900000 AM      04-MAY-22 04.51.22.179000 PM           3
         97209380 workflows.CheckSendAdministrativeStatusUsagepoint             03-MAY-22 09.49.53.563000 AM      04-MAY-22 04.54.23.002000 PM           3
         96862452 workflows.WaitComTaskToCompleteWorkflow                       25-APR-22 04.31.06.964000 AM      05-MAY-22 07.56.01.447000 AM           3
         96852273 workflows.WaitComTaskToCompleteWorkflow                       25-APR-22 03.21.19.303000 AM      08-MAY-22 04.07.43.157000 AM           2

7 rows selected.

Elapsed: 00:00:00.01
11:58:21 PENEXIS>
```

Now I think of it: even the “WHERE ERRORCOUNT > 0.0” condition is no longer needed as

2. Second alternative: as EXECUTIONERRORINFO is a very small table, make this the driving table. After all, we are only interested in the cases that have an ERRORCOUNT > 0.

```
SELECT PROCESSINSTANCEID, PROCESSID, START_DATE, END_DATE, STATUS, PARENTPROCESSINSTANCEID, OUTCOME, DURATION,
 USER_IDENTITY, PROCESSVERSION, PROCESSNAME, CORRELATIONKEY, EXTERNALID, PROCESSINSTANCEDESCRIPTION, SLA_DUE_DATE,
SLACOMPLIANCE, LASTMODIFICATIONDATE, ERRORCOUNT
 FROM
 ( SELECT LOG.PROCESSINSTANCEID, LOG.PROCESSID, LOG.START_DATE, LOG.END_DATE, LOG.STATUS, LOG.PARENTPROCESSINSTANCEID,   LOG.OUTCOME,
LOG.DURATION, LOG.USER_IDENTITY, LOG.PROCESSVERSION, LOG.PROCESSNAME, LOG.CORRELATIONKEY, LOG.EXTERNALID,
LOG.PROCESSINSTANCEDESCRIPTION, LOG.SLA_DUE_DATE, LOG.SLACOMPLIANCE, COALESCE ( INFO.LASTMODIFICATIONDATE,   LOG.END_DATE ) AS
LASTMODIFICATIONDATE,
 ( SELECT COUNT ( ERRINFO.ID )
```



```
 FROM EXECUTIONERRORINFO ERRINFO
 WHERE ERRINFO.PROCESS_INST_ID=LOG.PROCESSINSTANCEID   AND
ERRINFO.ERROR_ACK=0 ) AS ERRORCOUNT   FROM PROCESSINSTANCELOG LOG
 --
 LEFT JOIN PROCESSINSTANCEINFO INFO
 ON INFO.INSTANCEID=LOG.PROCESSINSTANCEID
 --
 WHERE LOG.PROCESSINSTANCEID
 in (
 select PROCESS_INST_ID
 from EXECUTIONERRORINFO
 where ERROR_ACK=0
 )
 --
 ) "dbSQL"
 ORDER BY START_DATE DESC FETCH FIRST 10 ROWS ONLY
/
```

Same response time: close to nothing.