

TP DBA report

Nassane marwa
siw G2

1. Create a new tablespace with a size of 4GB

sqlplus as / sysdba

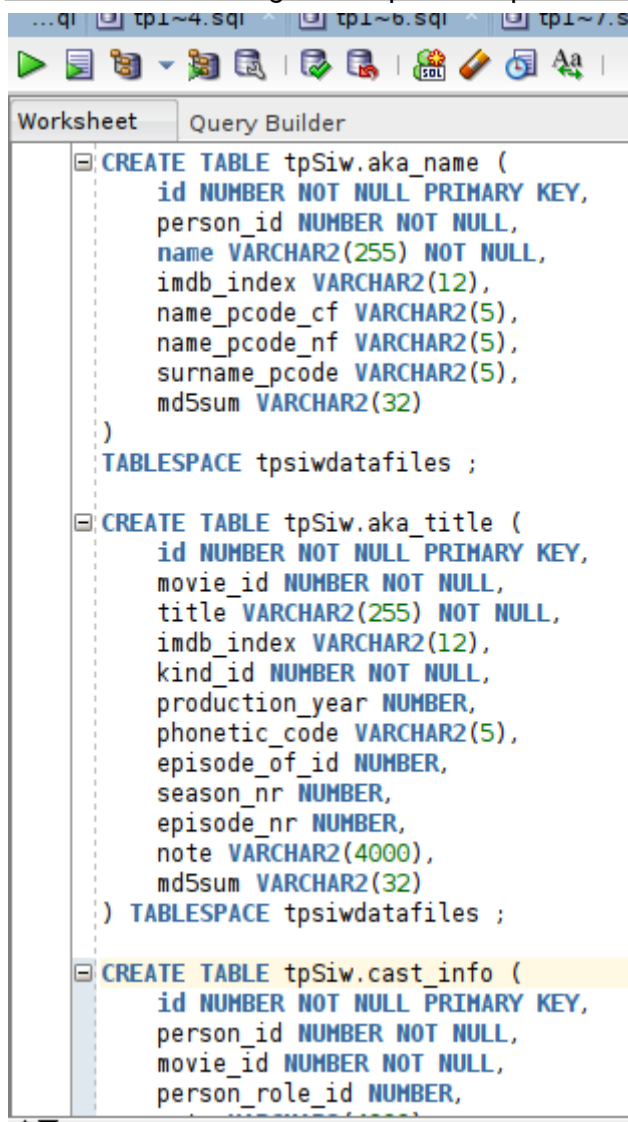
```
CREATE TABLESPACE tpsiwdatafiles DATAFILE '/opt/oracle/oradata/XE/XEPDB1/siw.dbf' size 4G
```

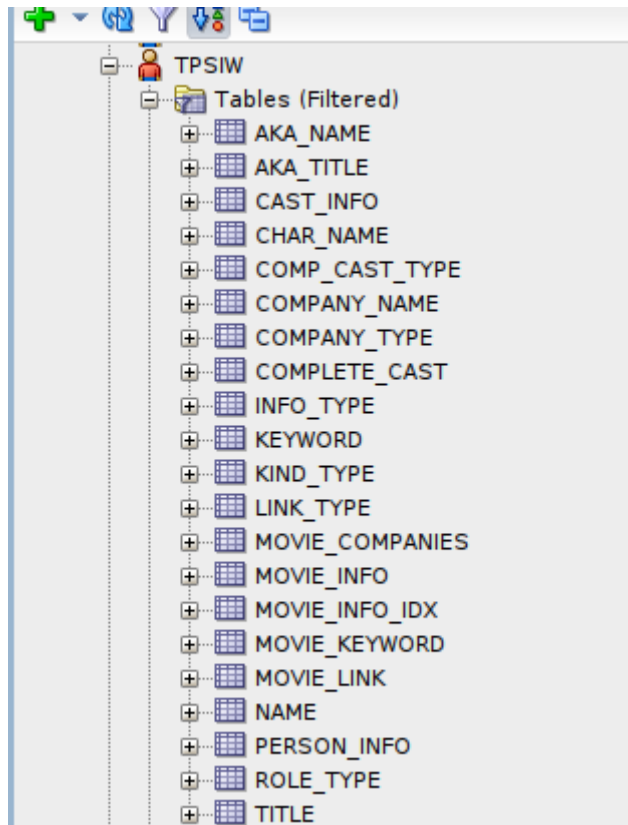
2. create a new user and associate them with both the DBA and Advisor roles

```
CREATE USER tpSiw IDENTIFIED BY 123
DEFAULT TABLESPACE tpsiwdatafiles
TEMPORARY TABLESPACE temp;
```

```
GRANT DBA TO tpSiw;
GRANT ADVISOR TO tpSiw;
commit;
```

3. create the table using the script and import the csv file





4. create sql tuning set

set the variables:

```
SET SERVEROUTPUT ON;
```

```
VARIABLE task_id NUMBER;
```

```
VARIABLE task_name VARCHAR2(255);
```

```
VARIABLE workload_name VARCHAR2(255);
```

```
EXECUTE :workload_name := 'workload';
```

```
EXECUTE DBMS_SQLTUNE.CREATE_SQLSET(:workload_name, 'description');
```

//crete the workload table

```
DROP TABLE user_workload;
```

```
CREATE TABLE workload
```

```
(
```

```
  username      varchar2(128), /* User who executes statement */
```

```
  module        varchar2(64),  /* Application module name */
```

```
  action        varchar2(64),  /* Application action name */
```

```
  elapsed_time   number,       /* Elapsed time for query */
```

```
  cpu_time       number,       /* CPU time for query */
```

```
  buffer_gets    number,       /* Buffer gets consumed by query */
```

```
  disk_reads     number,       /* Disk reads consumed by query */
```

```
  rows_processed number,       /* # of rows processed by query */
```

```
  executions     number,       /* # of times query executed */
```

```
  optimizer_cost number,       /* Optimizer cost for query */
```

```
  priority       number,       /* User-priority (1,2 or 3) */
```

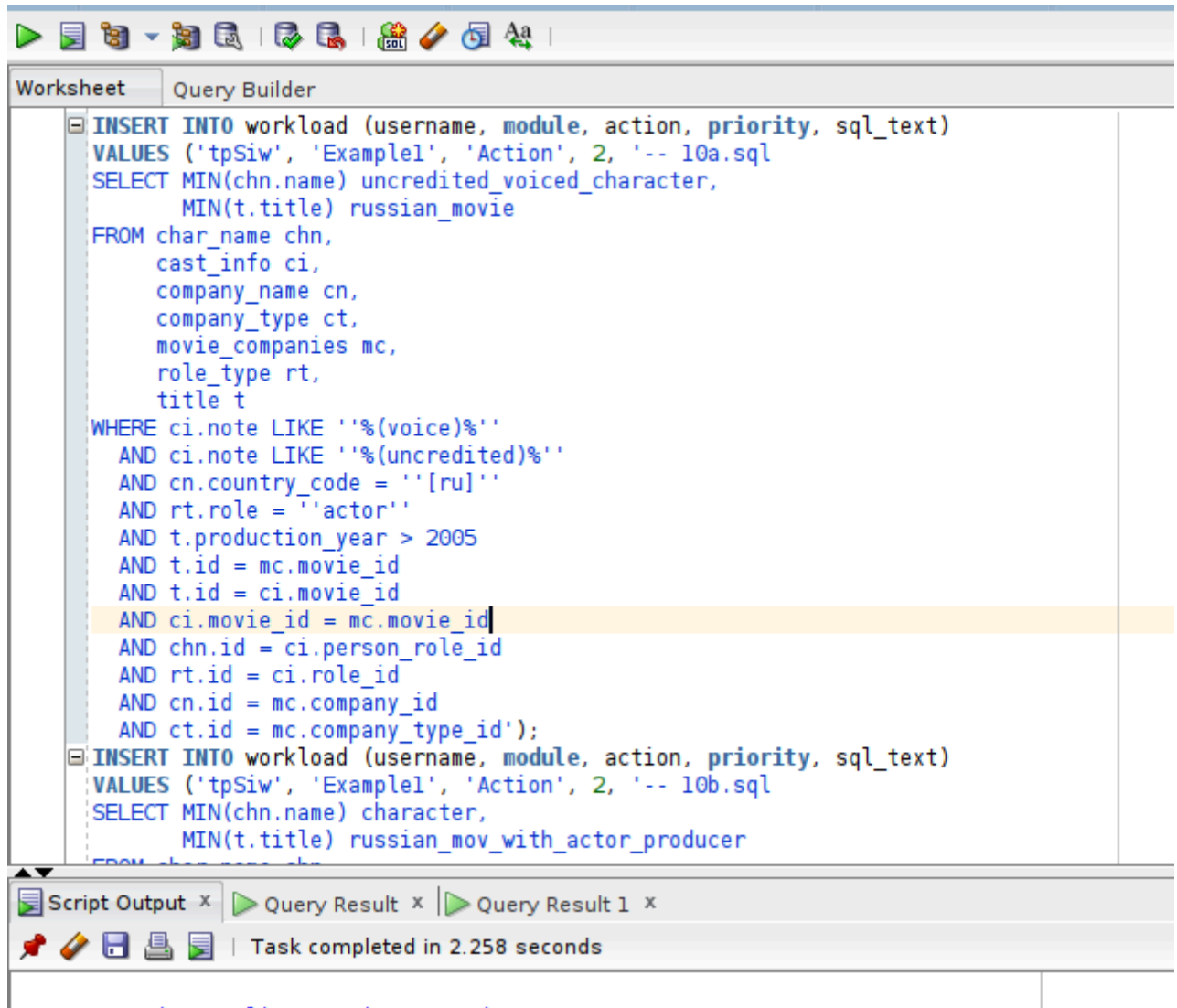
```
  last_execution_date date,    /* Last time query executed */
```

```
  stat_period    number,       /* Window exec time in seconds */
```

```
  sql_text       clob          /* Full SQL Text */
```

);

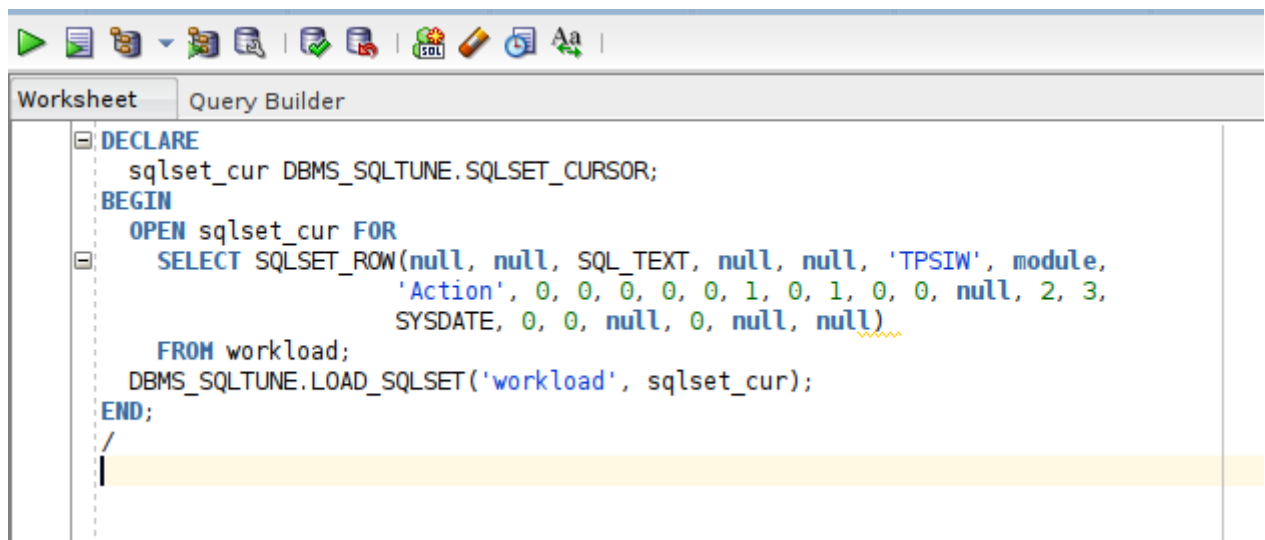
5. Load the created workload with the set of queries in the script



The screenshot shows the SQL Developer Query Builder interface. The 'Query Builder' tab is active, displaying two SQL queries being inserted into a table named 'workload'. The first query, labeled '10a.sql', inserts a record with 'tpSiw' as the username, 'Example1' as the module, 'Action' as the action, and a priority of 2. The SQL text for this query is a SELECT statement that finds the minimum character name and Russian movie title based on several criteria: the character's name must contain '(voice)', the note must contain '(uncredited)', the company code must be '[ru]', the role must be 'actor', the production year must be greater than 2005, and the character must be associated with a Russian movie. The second query, labeled '10b.sql', inserts a record with 'tpSiw' as the username, 'Example1' as the module, 'Action' as the action, and a priority of 2. The SQL text for this query is a SELECT statement that finds the minimum character name and Russian movie title based on several criteria: the character's name must contain '(voice)', the note must contain '(uncredited)', the company code must be '[ru]', the role must be 'actor', the production year must be greater than 2005, and the character must be associated with a Russian movie. The status bar at the bottom indicates 'Task completed in 2.258 seconds'.

```
INSERT INTO workload (username, module, action, priority, sql_text)
VALUES ('tpSiw', 'Example1', 'Action', 2, '-- 10a.sql
SELECT MIN(chn.name) uncredited_voiced_character,
       MIN(t.title) russian_movie
FROM char_name chn,
     cast_info ci,
     company_name cn,
     company_type ct,
     movie_companies mc,
     role_type rt,
     title t
WHERE ci.note LIKE '%(voice)%'
      AND ci.note LIKE '%(uncredited)%'
      AND cn.country_code = '[ru]'
      AND rt.role = 'actor'
      AND t.production_year > 2005
      AND t.id = mc.movie_id
      AND t.id = ci.movie_id
      AND ci.movie_id = mc.movie_id
      AND chn.id = ci.person_role_id
      AND rt.id = ci.role_id
      AND cn.id = mc.company_id
      AND ct.id = mc.company_type_id');
INSERT INTO workload (username, module, action, priority, sql_text)
VALUES ('tpSiw', 'Example1', 'Action', 2, '-- 10b.sql
SELECT MIN(chn.name) character,
       MIN(t.title) russian_mov_with_actor_producer
FROM char_name chn,
     cast_info ci,
     company_name cn,
     company_type ct,
     movie_companies mc,
     role_type rt,
     title t
WHERE ci.note LIKE '%(voice)%'
      AND ci.note LIKE '%(uncredited)%'
      AND cn.country_code = '[ru]'
      AND rt.role = 'actor'
      AND t.production_year > 2005
      AND t.id = mc.movie_id
      AND t.id = ci.movie_id
      AND ci.movie_id = mc.movie_id
      AND chn.id = ci.person_role_id
      AND rt.id = ci.role_id
      AND cn.id = mc.company_id
      AND ct.id = mc.company_type_id');
```

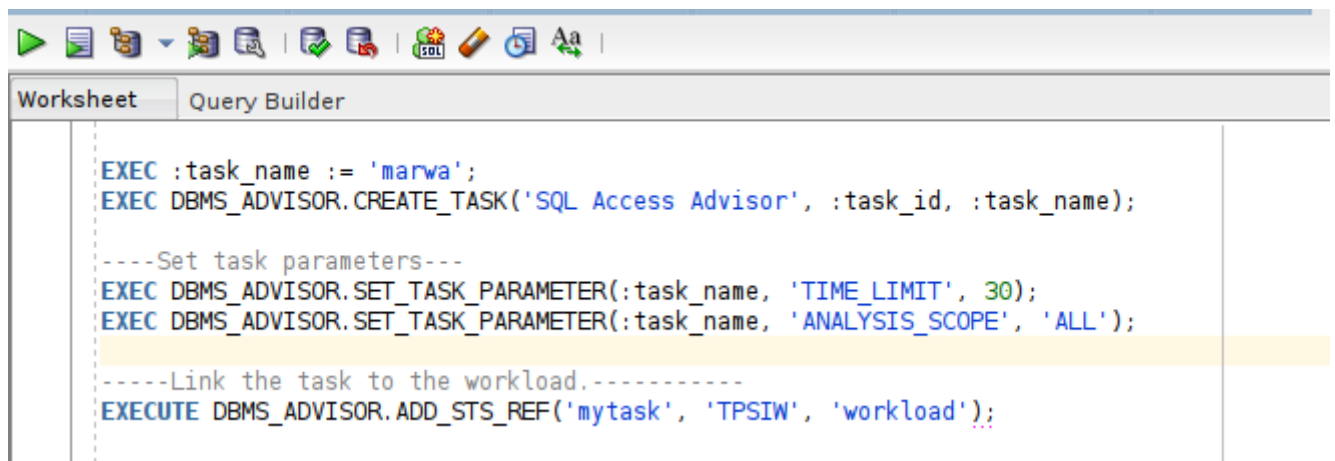
6. Declares a cursor to hold the rows from the workload table



The screenshot shows the SQL Developer Query Builder interface. The 'Query Builder' tab is active, displaying a SQL script. The script declares a cursor named 'sqlset_cur' of type 'DBMS_SQLTUNE.SQLSET_CURSOR'. It then begins a block where it opens the cursor for a SELECT statement. The SELECT statement is a complex query that selects various parameters for the cursor, including 'SQL_TEXT', 'module', 'action', and several numeric values. The query is executed against the 'workload' table. The script ends with 'END;' and a forward slash. The status bar at the bottom indicates 'Task completed in 2.258 seconds'.

```
DECLARE
sqlset_cur DBMS_SQLTUNE.SQLSET_CURSOR;
BEGIN
OPEN sqlset_cur FOR
SELECT SQLSET_ROW(null, null, SQL_TEXT, null, null, 'TPSIW', module,
                  'Action', 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, null, 2, 3,
                  SYSDATE, 0, 0, null, 0, null, null)
FROM workload;
DBMS_SQLTUNE.LOAD_SQLSET('workload', sqlset_cur);
END;
/
```

7. Creating and Configuring a SQL Access Advisor Task

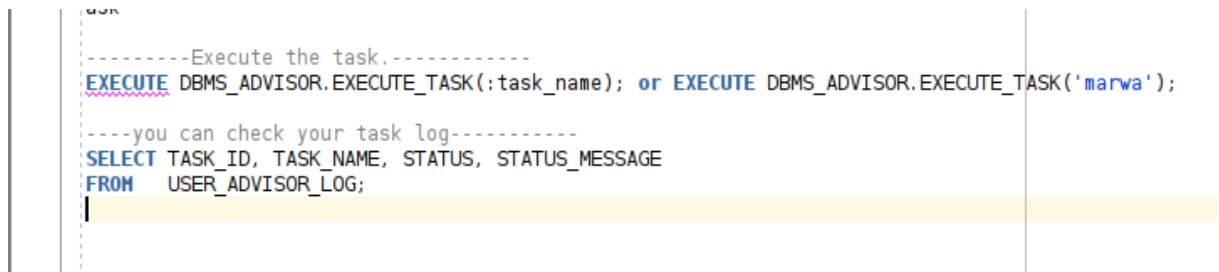


```
EXEC :task_name := 'marwa';
EXEC DBMS_ADVISOR.CREATE_TASK('SQL Access Advisor', :task_id, :task_name);

----Set task parameters---
EXEC DBMS_ADVISOR.SET_TASK_PARAMETER(:task_name, 'TIME_LIMIT', 30);
EXEC DBMS_ADVISOR.SET_TASK_PARAMETER(:task_name, 'ANALYSIS_SCOPE', 'ALL');

-----Link the task to the workload.-----
EXECUTE DBMS_ADVISOR.ADD_STS_REF('mytask', 'TPSIW', 'workload');
```

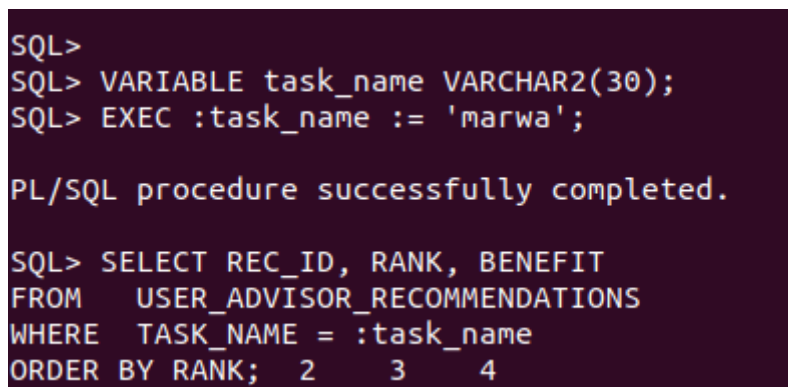
8. Executing a SQL Access Advisor Task



```
-----Execute the task.-----
EXECUTE DBMS_ADVISOR.EXECUTE_TASK(:task_name); or EXECUTE DBMS_ADVISOR.EXECUTE_TASK('marwa');

----you can check your task log-----
SELECT TASK_ID, TASK_NAME, STATUS, STATUS_MESSAGE
FROM USER_ADVISOR_LOG;
```

9. Viewing SQL Access Advisor Task Results:will contain recommendations generated by the SQL Access Advisor task



```
SQL>
SQL> VARIABLE task_name VARCHAR2(30);
SQL> EXEC :task_name := 'marwa';

PL/SQL procedure successfully completed.

SQL> SELECT REC_ID, RANK, BENEFIT
FROM USER_ADVISOR_RECOMMENDATIONS
WHERE TASK_NAME = :task_name
ORDER BY RANK; 2 3 4
```

the out put

REC_ID	RANK	BENEFIT
1	1	36146
2	2	122952
3	3	121996
4	4	107356
5	5	50428
6	6	123275
7	7	36036
8	8	39048
9	9	36047
10	10	29520
11	11	29674
REC_ID	RANK	BENEFIT
12	12	87414
13	13	98740
14	14	91935
15	15	87916
16	16	87671
17	17	87536
18	18	80544
19	19	80202
20	20	76447
21	21	76287
22	22	75904

```

      REC_ID      RANK      BENEFIT
-----
      23         23      18796
      24         24      17986
      25         25      58184
      26         26      68692
      27         27      58009
      28         28         2
      29         29         4
      30         30      20584
      31         31      23718
      32         32      19732
      33         33       7524

      REC_ID      RANK      BENEFIT
-----
      34         34      29116
      35         35      29092
      36         36      29092
      37         37         0
      38         38         1
      39         39      10290
      40         40       7500
      41         41       7142
      42         42       7229
      43         43       1229

43 rows selected.

SQL> 
```