## Task 01: Export Geo Location Reference

CREATE TABLESPACE sb\_mbackup DATAFILE 'DATA1.dbf' SIZE 100 M LOGGING;

GRANT CREATE SESSION TO sb\_mbackup;

CREATE USER sb\_mbackup IDENTIFIED BY pass DEFAULT TABLESPACE sb\_mbackup QUOTA UNLIMITED ON sb\_mbackup;

GRANT SELECT ON u\_dw\_references.cu\_geo\_regions TO sb\_mbackup;

GRANT SELECT ON u\_dw\_references.cu\_geo\_parts TO sb\_mbackup;

GRANT SELECT ON u\_dw\_references.cu\_geo\_systems TO sb\_mbackup;

GRANT SELECT ON u\_dw\_references.cu\_cntr\_group\_systems TO sb\_mbackup;

GRANT SELECT ON u\_dw\_references.cu\_cntr\_groups TO sb\_mbackup;

GRANT SELECT ON u\_dw\_references.cu\_cntr\_sub\_groups TO sb\_mbackup;

GRANT SELECT ON u\_dw\_references.cu\_countries TO sb\_mbackup;

GRANT SELECT ON u\_dw\_references.w\_geo\_object\_links TO sb\_mbackup;

GRANT CREATE TABLE TO sb\_mbackup;

CREATE TABLE exp\_locations AS

SELECT country\_geo\_id

, cntr.country\_id

, cntr.country\_code\_a3

, cntr.region\_desc AS country\_desc

, NVL ( g\_region\_id, -99 ) AS region\_geo\_id

, NVL ( reg.src\_continent\_id, -99 ) AS region\_id

, NVL ( reg.region\_code, 'n.d.' ) AS region\_code

, NVL ( reg.region\_desc, 'n.d.' ) AS region\_desc

, NVL ( g\_part\_id, -99 ) AS part\_geo\_id

, NVL ( part.part\_id, -99 ) AS part\_id

, NVL ( part.part\_code, 'n.d.' ) AS part\_code

, NVL ( part.part\_desc, 'n.d.' ) AS part\_desc

, NVL ( g\_system\_id, -99 ) AS geo\_system\_geo\_id

, NVL ( g\_sys.src\_geo\_system\_id, -99 ) AS geo\_system\_id

, NVL ( g\_sys.geo\_system\_code, 'n.d.' ) AS geo\_system\_code

, NVL ( g\_sys.geo\_system\_desc, 'n.d.' ) AS geo\_system\_desc

, NVL ( grp\_sub\_gr, -99 ) AS sub\_group\_geo\_id

, NVL ( sub\_grp.sub\_group\_id, -99 ) AS sub\_group\_id

, NVL ( sub\_grp.sub\_group\_code, 'n.d.' ) AS sub\_group\_code

, NVL ( sub\_grp.sub\_group\_desc, 'n.d.' ) AS sub\_group\_desc

, NVL ( grp\_group, -99 ) AS group\_geo\_id

, NVL ( grp.GROUP\_ID, -99 ) AS GROUP\_ID

, NVL ( grp.group\_code, 'n.d.' ) AS group\_code

, NVL ( grp.group\_desc, 'n.d.' ) AS group\_desc

, NVL ( grp\_sys, -99 ) AS grp\_system\_geo\_id

, NVL ( grp\_sys.grp\_system\_id, -99 ) AS grp\_system\_id

, NVL ( grp\_sys.grp\_system\_code, 'n.d.' ) AS grp\_system\_code

, NVL ( grp\_sys.grp\_system\_desc, 'n.d.' ) AS grp\_system\_desc

FROM ( SELECT parent\_geo\_id AS geo\_id

, CONNECT\_BY\_ROOT child\_geo\_id AS country\_geo\_id

, link\_type\_id

FROM u\_dw\_references.w\_geo\_object\_links

START WITH child\_geo\_id IN (SELECT geo\_id

FROM u\_dw\_references.cu\_countries)

CONNECT BY PRIOR parent\_geo\_id = child\_geo\_id) PIVOT (SUM ( geo\_id )

FOR link\_type\_id

IN (1 AS g\_system\_id

, 2 AS g\_part\_id

, 3 AS g\_region\_id

, 4 AS grp\_sys

, 5 AS grp\_group

, 6 AS grp\_sub\_gr)) t1

LEFT OUTER JOIN u\_dw\_references.cu\_countries cntr

ON ( cntr.geo\_id = t1.country\_geo\_id )

LEFT OUTER JOIN u\_dw\_references.cu\_geo\_regions reg

ON ( reg.geo\_id = t1.g\_region\_id )

LEFT OUTER JOIN u\_dw\_references.cu\_geo\_parts part

ON ( part.geo\_id = t1.g\_part\_id )

LEFT OUTER JOIN u\_dw\_references.cu\_geo\_systems g\_sys

ON ( g\_sys.geo\_id = t1.g\_system\_id )

LEFT OUTER JOIN u\_dw\_references.cu\_cntr\_group\_systems grp\_sys

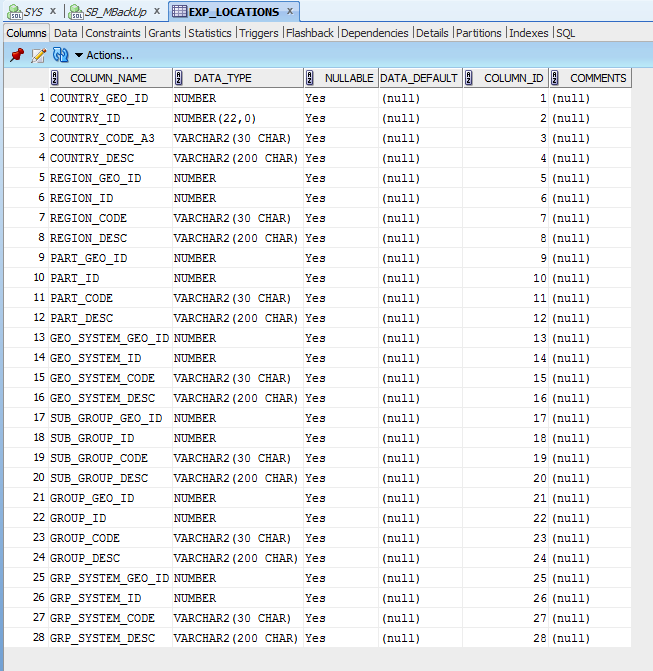
ON ( grp\_sys.geo\_id = t1.grp\_sys )

LEFT OUTER JOIN u\_dw\_references.cu\_cntr\_groups grp

ON ( grp.geo\_id = t1.grp\_group )

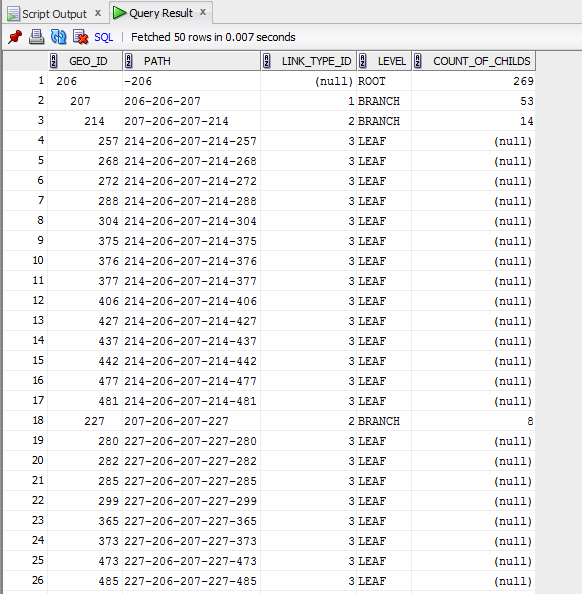
LEFT OUTER JOIN u\_dw\_references.cu\_cntr\_sub\_groups sub\_grp

ON ( sub\_grp.geo\_id = t1.grp\_sub\_gr );



**Required points:**

* Add Additional Columns to table:
  + Geo\_id types: Branch, ROOT, Leaf
  + Count of childs of Branch or Root, for Leafs this Field you have fill by NULL
  + Full path of Dependencies by Example: ROOT -> BRANCH -> BRANCH -> LEAF
* SELECT LPAD ( ' '
* , LEVEL \* 2 - 1
* , ' ' )
* || child\_geo\_id
* AS geo\_id
* , parent\_geo\_id
* || SYS\_CONNECT\_BY\_PATH ( child\_geo\_id
* , '-' )
* AS PATH
* , link\_type\_id
* , CASE
* WHEN LEVEL = 1 THEN 'ROOT'
* WHEN CONNECT\_BY\_ISLEAF = 1 THEN 'LEAF'
* ELSE 'BRANCH'
* END
* AS "LEVEL"
* , CASE
* WHEN CONNECT\_BY\_ISLEAF = 1 THEN
* NULL
* ELSE
* ( SELECT COUNT ( child\_geo\_id )
* FROM u\_dw\_references.t\_geo\_object\_links
* START WITH parent\_geo\_id = t1.child\_geo\_id
* CONNECT BY PRIOR child\_geo\_id = parent\_geo\_id )
* END
* count\_of\_childs
* FROM (SELECT \*
* FROM u\_dw\_references.t\_geo\_object\_links
* UNION
* SELECT NULL
* , geo\_id
* , NULL
* FROM u\_dw\_references.lc\_cntr\_group\_systems
* UNION
* SELECT NULL
* , geo\_id
* , NULL
* FROM u\_dw\_references.lc\_geo\_systems) t1
* START WITH parent\_geo\_id IS NULL
* CONNECT BY PRIOR child\_geo\_id = parent\_geo\_id
* ORDER SIBLINGS BY child\_geo\_id;



## Task 02: Analyse Business hierarch Reference Analyses

**The Main Task** is to create hierarch analyses of any Dimension, according yours Solution Proposal and DWH Solution Concept from Module 6. Introduction to DWH

**Required points:**

* Create Denormalized table data using CONNENT\_BY
* Use START WITH Clause
* Use CONNECT\_BY\_ROOT to analyses any Branch levels
* Analyze Main Root Branch, and 2 Sub Branches

SELECT country\_geo\_id

, cntr.country\_id

, cntr.country\_code\_a3

, cntr.region\_desc AS country\_desc

, NVL ( g\_region\_id, -99 ) AS region\_geo\_id

, NVL ( reg.src\_continent\_id, -99 ) AS region\_id

, NVL ( reg.region\_code, 'n.d.' ) AS region\_code

, NVL ( reg.region\_desc, 'n.d.' ) AS region\_desc

, NVL ( g\_part\_id, -99 ) AS part\_geo\_id

, NVL ( part.part\_id, -99 ) AS part\_id

, NVL ( part.part\_code, 'n.d.' ) AS part\_code

, NVL ( part.part\_desc, 'n.d.' ) AS part\_desc

FROM ( SELECT parent\_geo\_id AS geo\_id

, CONNECT\_BY\_ROOT child\_geo\_id AS country\_geo\_id

, link\_type\_id

FROM u\_dw\_references.w\_geo\_object\_links

START WITH child\_geo\_id IN (SELECT geo\_id

FROM u\_dw\_references.cu\_countries)

CONNECT BY PRIOR parent\_geo\_id = child\_geo\_id) PIVOT (SUM ( geo\_id )

FOR link\_type\_id

IN (2 AS g\_part\_id, 3 AS g\_region\_id)) t1

LEFT OUTER JOIN u\_dw\_references.cu\_countries cntr

ON ( cntr.geo\_id = t1.country\_geo\_id )

LEFT OUTER JOIN u\_dw\_references.cu\_geo\_regions reg

ON ( reg.geo\_id = t1.g\_region\_id )

LEFT OUTER JOIN u\_dw\_references.cu\_geo\_parts part

ON ( part.geo\_id = t1.g\_part\_id );