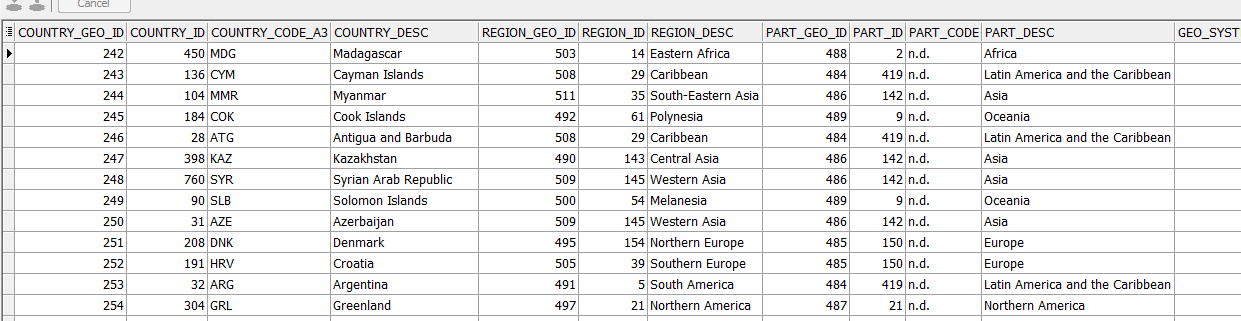
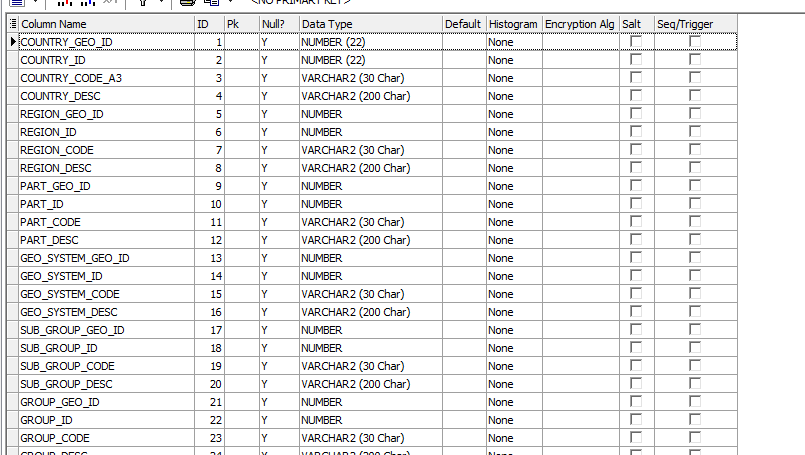
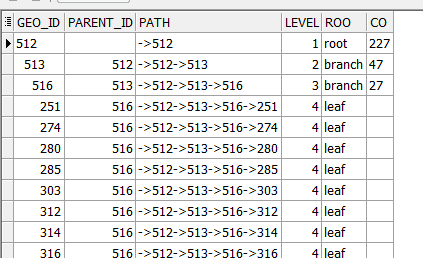
## Task 01: Export Geo Location Reference

|  |
| --- |
| CREATE TABLE exp\_country AS  SELECT lcc.geo\_id AS country\_geo\_id  , lcc.country\_id  , lcc.country\_code\_a3  , lcc.country\_desc AS country\_desc  , NVL ( lcr.geo\_id, -99 ) AS region\_geo\_id  , NVL ( lcr.region\_id, -99 ) AS region\_id  , NVL ( lcr.region\_code, 'n.d.' ) AS region\_code  , NVL ( lcr.region\_desc, 'n.d.' ) AS region\_desc  , NVL ( lcp.geo\_id, -99 ) AS part\_geo\_id  , NVL ( lcp.part\_id, -99 ) AS part\_id  , NVL ( lcp.part\_code, 'n.d.' ) AS part\_code  , NVL ( lcp.part\_desc, 'n.d.' ) AS part\_desc  , NVL ( lcs.geo\_id, -99 ) AS geo\_system\_geo\_id  , NVL ( lcs.geo\_id, -99 ) AS geo\_system\_id  , NVL ( lcs.geo\_system\_code, 'n.d.' ) AS geo\_system\_code  , NVL ( lcs.geo\_system\_desc, 'n.d.' ) AS geo\_system\_desc  , NVL ( lcsg.geo\_id, -99 ) AS sub\_group\_geo\_id  , NVL ( lcsg.sub\_group\_id, -99 ) AS sub\_group\_id  , NVL ( lcsg.sub\_group\_code, 'n.d.' ) AS sub\_group\_code  , NVL ( lcsg.sub\_group\_desc, 'n.d.' ) AS sub\_group\_desc  , NVL ( lcg.geo\_id, -99 ) AS group\_geo\_id  , NVL ( lcg.GROUP\_ID, -99 ) AS GROUP\_ID  , NVL ( lcg.group\_code, 'n.d.' ) AS group\_code  , NVL ( lcg.group\_desc, 'n.d.' ) AS group\_desc  , NVL ( lcgs.geo\_id, -99 ) AS grp\_system\_geo\_id  , NVL ( lcgs.grp\_system\_id, -99 ) AS grp\_system\_id  , NVL ( lcgs.grp\_system\_code, 'n.d.' ) AS grp\_system\_code  , NVL ( lcgs.grp\_system\_desc, 'n.d.' ) AS grp\_system\_desc  FROM (SELECT \*  FROM ( SELECT parent\_geo\_id AS geo\_id  , CONNECT\_BY\_ROOT child\_geo\_id AS roo  , link\_type\_id  FROM t\_geo\_object\_links gol  START WITH child\_geo\_id IN (SELECT geo\_id  FROM lc\_countries)  CONNECT BY PRIOR parent\_geo\_id = child\_geo\_id) PIVOT (SUM ( geo\_id )  FOR link\_type\_id  IN (1 AS l\_1  , 2 AS l\_2  , 3 AS l\_3  , 4 AS l\_4  , 5 AS l\_5  , 6 AS l\_6))) piv  LEFT JOIN lc\_countries lcc  ON piv.roo = lcc.geo\_id  LEFT JOIN lc\_geo\_regions lcr  ON lcr.geo\_id = piv.l\_3  LEFT JOIN lc\_geo\_parts lcp  ON lcp.geo\_id = piv.l\_2  LEFT JOIN lc\_geo\_systems lcs  ON piv.l\_1 = lcs.geo\_id  LEFT JOIN lc\_cntr\_sub\_groups lcsg  ON piv.l\_4 = lcsg.geo\_id  LEFT JOIN lc\_cntr\_groups lcg  ON piv.l\_5 = lcg.geo\_id  LEFT JOIN lc\_cntr\_group\_systems lcgs  ON piv.l\_6 = lcgs.geo\_id  ORDER BY roo; |





CONNECT BY AND ADDITIONAL COLOUMNS



|  |
| --- |
| SELECT LPAD ( ' '  , 2 \* ( LEVEL - 1 ) )  || child\_geo\_id  AS geo\_id  , parent\_geo\_id AS parent\_id  , SYS\_CONNECT\_BY\_PATH ( child\_geo\_id  , '->' )  AS PATH  , LEVEL  , CASE  WHEN CONNECT\_BY\_ROOT child\_geo\_id = child\_geo\_id THEN 'root'  WHEN CONNECT\_BY\_ISLEAF = 1 THEN 'leaf'  ELSE 'branch'  END  AS roo  , DECODE ( CONNECT\_BY\_ISLEAF  , 1, NULL  , ( SELECT COUNT ( child\_geo\_id )  FROM t\_geo\_object\_links  START WITH parent\_geo\_id = gol.child\_geo\_id  CONNECT BY PRIOR child\_geo\_id = parent\_geo\_id ) )  AS co  FROM (SELECT \*  FROM t\_geo\_object\_links  UNION ALL  SELECT NULL  , geo\_id  , NULL  FROM lc\_cntr\_group\_systems) gol  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

|  |
| --- |
| SELECT lcc.geo\_id AS country\_geo\_id  , lcc.country\_id  , lcc.country\_code\_a3  , lcc.country\_desc AS country\_desc  , NVL ( lcr.geo\_id, -99 ) AS region\_geo\_id  , NVL ( lcr.region\_id, -99 ) AS region\_id  , NVL ( lcr.region\_desc, 'n.d.' ) AS region\_desc  , NVL ( lcp.geo\_id, -99 ) AS part\_geo\_id  , NVL ( lcp.part\_id, -99 ) AS part\_id  , NVL ( lcp.part\_desc, 'n.d.' ) AS part\_desc  FROM (SELECT \*  FROM ( SELECT parent\_geo\_id AS geo\_id  , CONNECT\_BY\_ROOT child\_geo\_id AS roo  , link\_type\_id  FROM t\_geo\_object\_links gol  START WITH child\_geo\_id IN (SELECT geo\_id  FROM lc\_countries)  CONNECT BY PRIOR parent\_geo\_id = child\_geo\_id) PIVOT (SUM ( geo\_id )  FOR link\_type\_id  IN (2 AS l\_2  , 3 AS l\_3  ))) piv  LEFT JOIN lc\_countries lcc  ON piv.roo = lcc.geo\_id AND lcc.localization\_id = (:loc\_param)  LEFT JOIN lc\_geo\_regions lcr  ON lcr.geo\_id = piv.l\_3 AND lcr.localization\_id = (:loc\_param)  LEFT JOIN lc\_geo\_parts lcp  ON lcp.geo\_id = piv.l\_2 AND lcp.localization\_id = (:loc\_param)  ORDER BY roo; |

**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

**Task Results:**

Create required objects:

* + Prepare Document with Screenshot of analyses Data result
  + Prepare script and put it to GIT