



UCOP Data Users Group

March 9th, 2020

OLAP Functions

- ▶ OLAP (Online Analytical Processing) functions allow you to flexibly create subgroups in your query by partition the data sets.
- ▶ It can be used in the SELECT statements in SQL to effectively perform arithmetic, analytical, ranking functions, find lead and lag values, cumulative values by subgroups of the entire data set.
- ▶ It simplifies the SQL query significantly.

OLAP Function diagram

PARTITION BY department

ORDER BY startdate

RANK()

firstname	department	startdate
Andrew	1	1/23/1999
Jacob	1	7/11/1990
Daniel	2	6/24/2004
Anna	1	10/7/2001
Pierre	1	2/22/2009
Ruth	2	6/6/1998
Anthony	1	11/29/1995
Isabella	2	9/28/1997
Jose	2	3/17/2013

firstname	department	startdate
Andrew	1	1/23/1999
Jacob	1	7/11/1990
Anna	1	10/7/2001
Pierre	1	2/22/2009
Anthony	1	11/29/1995

firstname	department	startdate
Jacob	1	7/11/1990
Anthony	1	11/29/1995
Andrew	1	1/23/1999
Anna	1	10/7/2001
Pierre	1	2/22/2009

firstname	department	startdate	rank
Jacob	1	7/11/1990	1
Anthony	1	11/29/1995	2
Andrew	1	1/23/1999	3
Anna	1	10/7/2001	4
Pierre	1	2/22/2009	5

firstname	department	startdate
Ruth	2	6/6/1998
Daniel	2	6/24/2004
Jose	2	3/17/2013
Isabella	2	9/28/1997

firstname	department	startdate
Isabella	2	9/28/1997
Daniel	2	6/24/2004
Jose	2	3/17/2013
Ruth	2	6/6/2013

firstname	department	startdate	rank
Isabella	2	9/28/1997	1
Daniel	2	6/24/2004	2
Jose	2	3/17/2013	3
Ruth	2	6/6/2013	4

Syntax

- ▶ **SELECT** **Function**(*{ c1 }*) **OVER** (**PARTITION BY** *c2, c3, ... { ORDER BY c4, c5... }*) **AS** *Oc1, c2, c3, c6, c7, ...* **FROM** T1
 - T1 has columns c1, c2, c3, c4, c5, c6, c7.
- ▶ **OLAP Functions:**
 - ROW_NUMBER, RANK, DENSE_RANK
 - FIRST_VALUE, LAST_VALUE, NTH_VALUE
 - SUM, AVG, MAX, MIN, COUNT
 - LEAD, LAG

Exercise Goal – Enrollment Star

- ▶ Write a SQL query to select the following
 1. EOT if available or 3WK other wise
 2. Find the earliest, latest year and term enrolled
 3. Cumulative total number of fall terms enrolled
 4. Set flag to 1 whenever the student level is changed, else set to 0

OLAP Functions used

1. EOT if available or 3WK other wise – `ROW_NUMBER()`
2. Find the earliest, latest year and term enrolled – `FIRST_VALUE()`

Technical Requirements

► Query format:

```
SELECT *, F_ACAD_YEAR, F_TERM_CD, L_ACAD_YEAR, L_TERM_CD  
FROM
```

```
(
```

Query 2

```
SELECT *  
FROM
```

```
(
```

Query 1

```
SELECT cmp_stud_id, acad_yr, term, stud_lvl, ROW_NUMBER() as FROM  
Enrollment_fact and dimensions with where clause
```

```
) WHERE ROWNUM = 1 --Selects EOT if available, otherwise 3WK
```

```
)
```

Execution steps

▶ Base Query:

- ❑ Write query from enrollment star using enrollment fact and associated dimensions.
- ❑ Create a column CMP_STUD_ID (campus code||student id)

▶ QUERY1:

- ❑ Use ROW_NUMBER() partition by CMP_STUD_ID, ACAD_YR, TERM order by RECORD_TYPE desc
- ❑ Filter the rows where ROW_NUMBER = 1

▶ QUERY2:

- ❑ Use FIRST_VALUE() partition by CMP_STUD_ID order by ACAD_YR, TERM asc to determine the earliest year/term enrolled & order by ACAD_YR desc, TERM desc to determine the latest year/term enrolled
- ❑ Create columns F_ACAD_YR & F_TERM for earliest year/term enrolled
- ❑ Create columns L_ACAD_YR & L_TERM for latest year/term enrolled

Technical Requirements

- ▶ **Schema:**
 - STUD_BI
- ▶ **Table Name:**
 - ENROLLMENT_F
 - ACADEMIC_SUB_TERM_D
 - STUDENT_D
 - STUDENT_LEVEL_D
- ▶ **Column Names:**
 - Enrollment fact: Enrollment major component number
 - Academic sub term dimension: Academic year, term, record type
 - Student dimension: Student ID
 - Campus Location dimension: Campus code
 - Student Level dimension: Student level code
- ▶ **Row Filter Rules for the Base Query:**
 - Major component number = 1
 - Student level code in ('5', '6', '7', '8')
 - Stud ID in ('10000001') [Stud id here is a random number used for example purposes, it is not a real stud id]

Technical Requirements

- ▶ **BASE QUERY & QUERY 1** (*ROWNUM*)
 - ROW_NUMBER() OVER (PARTITION BY *cmp_stud_id*, *year*, *term* order by *rec_type* desc)

- ▶ **QUERY2** (*F_ACAD_YEAR*, *F_TERM_CD*)
 - FIRST_VALUE(*year*) OVER (PARTITION BY *cmp_stud_id* order by *year* asc, *term* asc)
 - FIRST_VALUE(*term*) OVER (PARTITION BY *cmp_stud_id* order by *year* asc, *term* asc)

- ▶ **QUERY2** (*L_ACAD_YEAR*, *L_TERM_CD*)
 - FIRST_VALUE(*year*) OVER (PARTITION BY *cmp_stud_id* order by *year* desc, *term* desc)
 - FIRST_VALUE(*term*) OVER (PARTITION BY *cmp_stud_id* order by *year* desc, *term* desc)

Expected results – Query

```

SELECT A_CMP_STUD_ID, A_ACAD_SUB_T_REC_TY, A_ACAD_YEAR, A_TERM_CD, A_STUD_LVL_CD, F_ACAD_YEAR, F_TERM_CD, L_ACAD_YEAR, L_TERM_CD
FROM
(
  SELECT *,
  --select earliest yr /term for every degree enrolled
  FIRST_VALUE(A_ACAD_YEAR) OVER(PARTITION BY A_CMP_STUD_ID order by A_ACAD_YEAR asc, A_TERM_CD asc) as F_ACAD_YEAR,
  --select earliest yr /term for every degree enrolled
  FIRST_VALUE(A_TERM_CD) OVER(PARTITION BY A_CMP_STUD_ID order by A_ACAD_YEAR asc, A_TERM_CD asc) as F_TERM_CD,
  --select latest yr /term for every degree enrolled
  FIRST_VALUE(A_ACAD_YEAR) OVER(PARTITION BY A_CMP_STUD_ID order by A_ACAD_YEAR desc, A_TERM_CD desc) as L_ACAD_YEAR,
  --select latest yr /term for every degree enrolled
  FIRST_VALUE(A_TERM_CD) OVER(PARTITION BY A_CMP_STUD_ID order by A_ACAD_YEAR desc, A_TERM_CD desc) as L_TERM_CD
  FROM
  (
    SELECT * FROM
    (
      --Select EOT if available, otherwise select 3WK record type
      SELECT *,ROW_NUMBER() OVER(PARTITION BY A_CMP_STUD_ID, A_ACAD_YEAR, A_TERM_CD order by A_ACAD_SUB_T_REC_TY desc) as ROWNUM
      FROM
      (
        SELECT
        ast_d.ACAD_SUB_T_REC_TY as A_ACAD_SUB_T_REC_TY, ast_d.ACAD_SUB_T_ACAD_YR as A_ACAD_YEAR, ast_d.ACAD_SUB_T_CD as A_TERM_CD,
        TRIM(cl_d.CMP_LOC_LOC1_CD||s_d.STUD_ID) as A_CMP_STUD_ID, s_d.STUD_LOC_CMP_CD as A_STUD_LOC_CMP_CD,
        TRIM(s_d.STUD_ID) as A_STUD_ID, sl_d.STUD_LVL_CD as A_STUD_LVL_CD
        FROM
        STUD_BI.ENROLLMENT_F e_f
        INNER JOIN STUD_BI.ACADEMIC_SUB_TERM_D ast_d ON e_f.ACAD_SUB_T_KEY = ast_d.ACAD_SUB_T_KEY
        INNER JOIN STUD_BI.STUDENT_D s_d ON e_f.STUD_KEY = s_d.STUD_KEY
        INNER JOIN STUD_BI.CAMPUS_LOCATION_D cl_d ON e_f.CMP_LOC_KEY = cl_d.CMP_LOC_KEY
        INNER JOIN STUD_BI.STUDENT_LEVEL_D sl_d ON e_f.STUD_LVL_KEY = sl_d.STUD_LVL_KEY
        WHERE
        --Select only the graduate degree students.
        sl_d.STUD_LVL_CD in ('5', '6', '7', '8', 'P')
        -- Select only the primary major record. This will ensure the grain of the query is one row per student, year, and term.
        AND e_f.ENRL_MAJ_CMPNT_NUM = 1
        --Stud id here is a random number used for example purposes, it is not a real stud id
        AND s_d.STUD_ID IN ('10000001')
      )
    ) WHERE ROWNUM = 1
  )
)

```

Expected results – Output

A_CMP_STUD_ID	A_ACAD_SUB_T_REC_TY	A_ACAD_YEAR	A_TERM_CD	A_STUD_LVL_CD	F_ACAD_YEAR	F_TERM_CD	L_ACAD_YEAR	L_TERM_CD
10000001	EOT	2006 3	5		2006 3		2012 3	
10000001	EOT	2006 4	5		2006 3		2012 3	
10000001	EOT	2007 2	5		2006 3		2012 3	
10000001	EOT	2007 3	5		2006 3		2012 3	
10000001	EOT	2007 4	5		2006 3		2012 3	
10000001	EOT	2008 2	5		2006 3		2012 3	
10000001	EOT	2008 3	6		2006 3		2012 3	
10000001	EOT	2008 4	6		2006 3		2012 3	
10000001	EOT	2009 2	6		2006 3		2012 3	
10000001	EOT	2009 3	6		2006 3		2012 3	
10000001	EOT	2009 4	6		2006 3		2012 3	
10000001	EOT	2010 2	6		2006 3		2012 3	
10000001	EOT	2010 3	7		2006 3		2012 3	
10000001	EOT	2010 4	7		2006 3		2012 3	
10000001	EOT	2011 2	7		2006 3		2012 3	
10000001	EOT	2011 3	7		2006 3		2012 3	
10000001	EOT	2011 4	7		2006 3		2012 3	
10000001	EOT	2012 2	7		2006 3		2012 3	
10000001	EOT	2012 3	7		2006 3		2012 3	

- ✓ Stud id here is a random number used for example purposes, it is not a real stud id.
- ✓ Query picks EOT is available, else 3WK is selected.
- ✓ A_ACAD_YR & A_TERM_CD is the year/term enrolled.
- ✓ F_ACAD_YR & F_TERM_CD is the earliest year/term enrolled.
- ✓ L_ACAD_YR & L_TERM_CD is the latest year/term enrolled.
- ✓ The grain of the query is one record per student, year and term enrolled.