ucdug

UCOP Data Users Group

November 28, 2018

Motivations

- GROUP BY allows you to aggregate your data at a certain level. But you can only have one grouping per query.
- What if you wanted to:

1. Have a rolling average (maybe due to small cell sizes)
2. Compare subgroups to larger groups
3. Count consecutive terms a student was enrolled

OLAP SQL Functions

- OLAP (Online Analytical Processing) functions allow you to flexibly create subgroups in your query
- Somewhat similar in concept to table calculations in Tableau
- DUG has previously covered some of these functions: ROW_NUMBER, RANK, DENSE_RANK

Order of operations

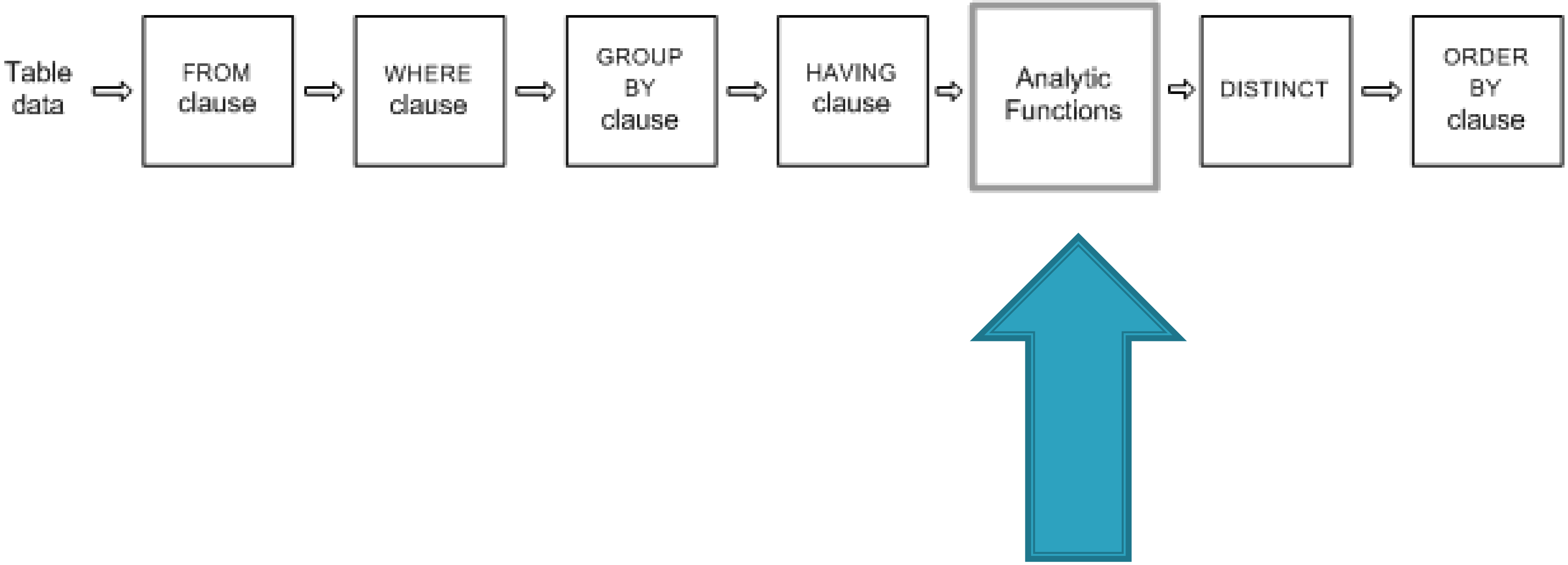
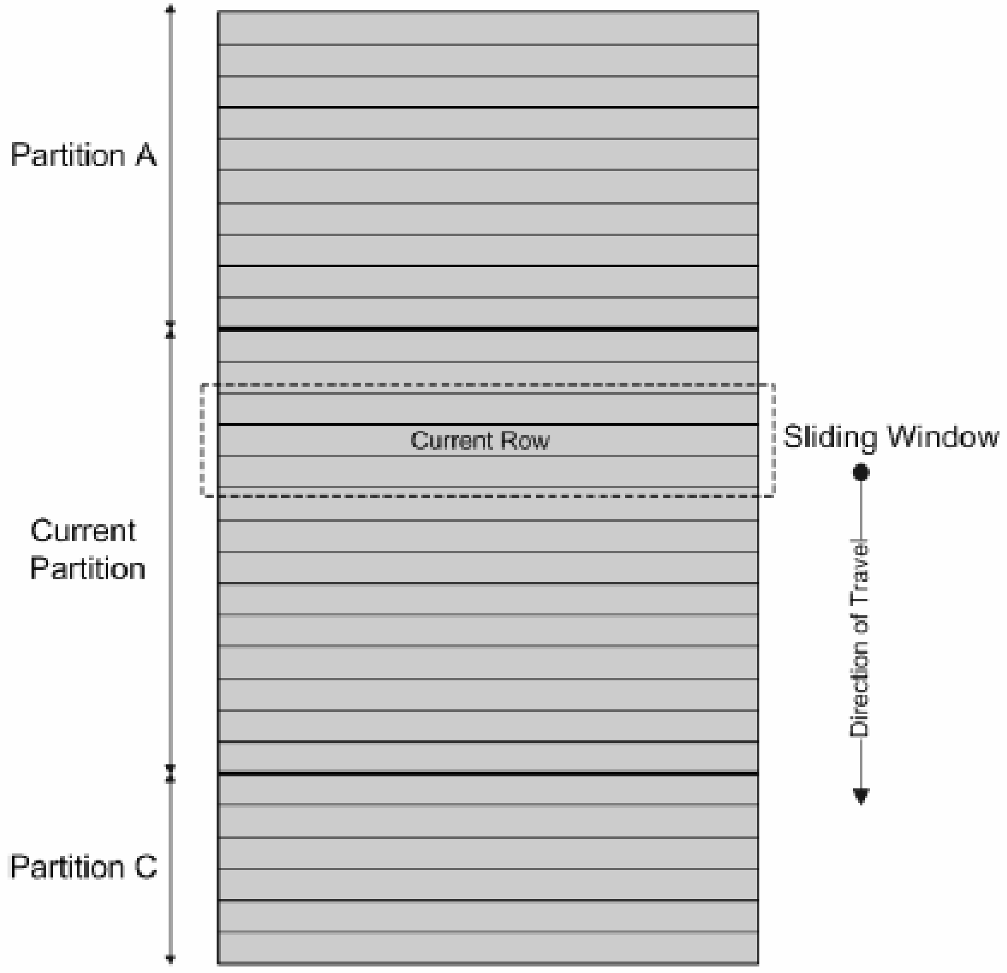


Table data ⇨ FROM clause ⇨ WHERE clause ⇨ GROUP BY clause ⇨ HAVING clause ⇨ Analytic Functions ⇨ DISTINCT ⇨ ORDER BY clause

Key Concepts

- PARTITIONS: subgroups in the data that you want to analyze OVER
- WINDOW FRAMES: moving frames inside each PARTITION



ucdug

Key Concept Examples

- DUG previously used PARTITIONS to create row numbers and ranks
- Three-year rolling average of graduation rates by ethnicity: Each ethnic group could be a PARTITION in your data. The WINDOW FRAME could be the prior year, the current year, and the following year. During the first year and last year, the WINDOW would only include two years.

Syntax

- [Function] OVER (PARTITION BY [A],[B] ORDER BY [C],[D] [Window Frame])
- Partition, order, and window frame are all optional
- Two types of window frames: ROWS (based on the actual order/number of rows) and RANGE (based on the variable you ORDER BY)

(This is for DB2, other databases may differ)

ROWS and RANGE examples

- ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
- ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
- ORDER BY year asc RANGE between CURRENT ROW and 3 FOLLOWING (this will include all rows with values for “year” that match the current row or the three years after that

Note:When using “RANGE”, you can only have one numeric variable in the ORDER BY

(This is for DB2, other databases may differ)

Example query

For fall 2017, show the likelihood of being first-generation by UC race/ethnic 6-category compared to the campus overall. For example, if 50% of group X is first-generation compared to 25% at the campus overall, then the likelihood would be 2.0. Limit the query to undergraduates.

This query does not use a WINDOW FRAME. The key is the two different partitions.

```
select distinct ACADEMIC_YR, CAMPUS_CD, ENR_UC_ETHN_6_CAT,
float(avg(CASE when PARENT_EDUCATION_LVL='No College' then 1.0 else 0 end)
OVER (Partition by ACADEMIC_YR, CAMPUS_CD, ENR_UC_ETHN_6_CAT))/
float(avg(CASE when PARENT_EDUCATION_LVL='No College' then 1.0 else 0 end)
OVER (Partition by ACADEMIC_YR, CAMPUS_CD)) AS FirstGenDiff
FROM IRAP_BI.ENROLLMENT_DM
WHERE STUD_LVL_UGR='Undergraduate' and ACADEMIC_YR=2017
```

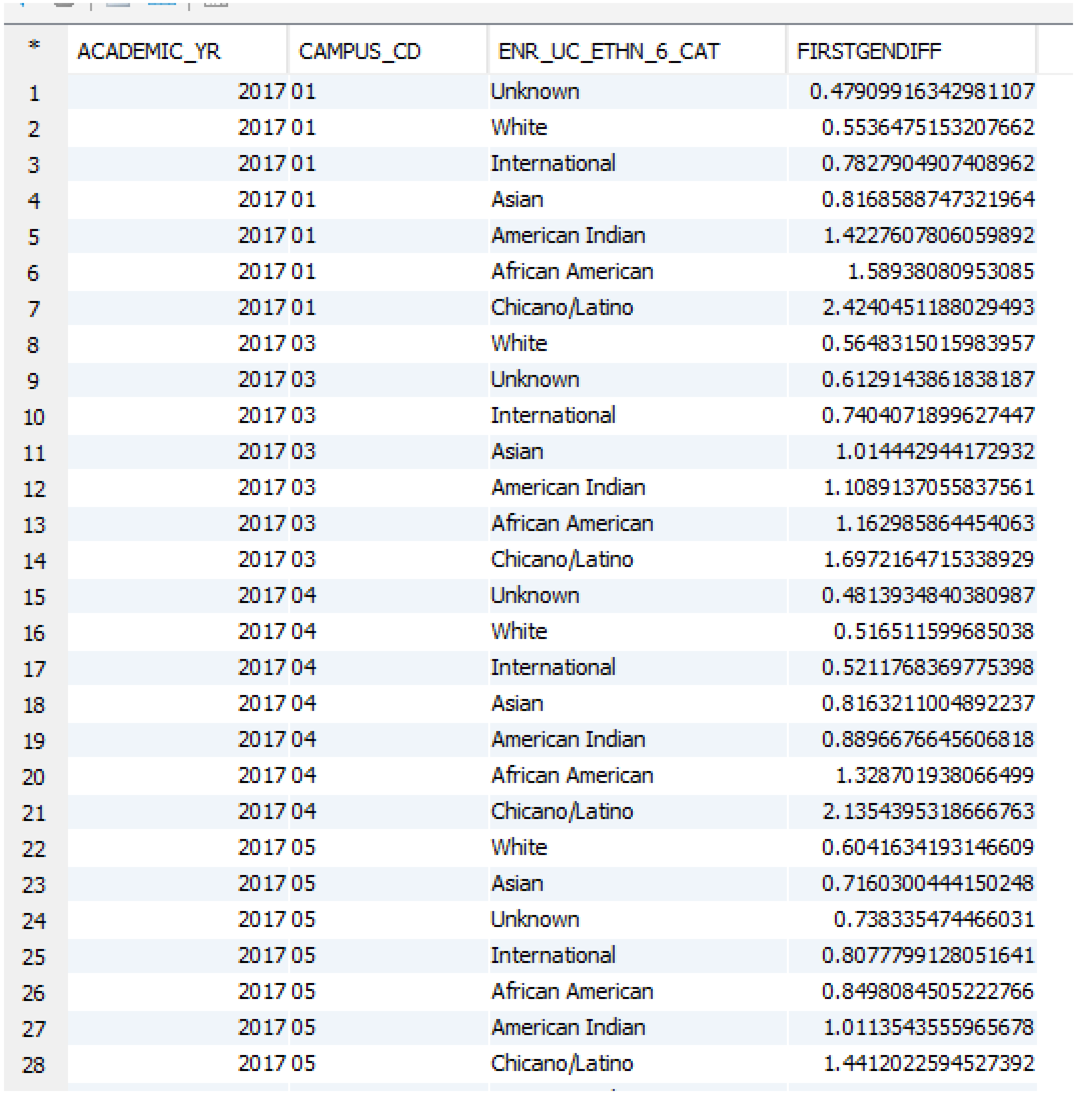
Code concepts

- Percent First Gen: **float**(**avg**(**CASE when** PARENT_EDUCATION_LVL='No College' **then** 1.0 **else** 0 **end**)
- Note the use of “float” and “1.0” to force DB2 not to round to whole numbers.

```
select distinct ACADEMIC_YR, CAMPUS_CD, ENR_UC_ETHN_6_CAT,
float(avg(CASE when PARENT_EDUCATION_LVL='No College' then 1.0
else 0 end) OVER (Partition by ACADEMIC_YR, CAMPUS_CD,
ENR_UC_ETHN_6_CAT))/
float(avg(CASE when PARENT_EDUCATION_LVL='No College' then 1.0
else 0 end) OVER (Partition by ACADEMIC_YR, CAMPUS_CD)) AS
FirstGenDiff
FROM IRAP_BI.ENROLLMENT_DM
WHERE STUD_LVL_UGR='Undergraduate' and ACADEMIC_YR=2017
```

ucdug

Result



	ACADEMIC_YR	CAMPUS_CD	ENR_UC_ETHN_6_CAT	FIRSTGENDIFF
1	2017	01	Unknown	0.47909916342981107
2	2017	01	White	0.5536475153207662
3	2017	01	International	0.7827904907408962
4	2017	01	Asian	0.8168588747321964
5	2017	01	American Indian	1.4227607806059892
6	2017	01	African American	1.58938080953085
7	2017	01	Chicano/Latino	2.4240451188029493
8	2017	03	White	0.5648315015983957
9	2017	03	Unknown	0.6129143861838187
10	2017	03	International	0.7404071899627447
11	2017	03	Asian	1.014442944172932
12	2017	03	American Indian	1.1089137055837561
13	2017	03	African American	1.162985864454063
14	2017	03	Chicano/Latino	1.6972164715338929
15	2017	04	Unknown	0.4813934840380987
16	2017	04	White	0.51651159968503 8
17	2017	04	International	0.5211768369775398
18	2017	04	Asian	0.8163211004892237
19	2017	04	American Indian	0.8896676645606818
20	2017	04	African American	1.328701938066499
21	2017	04	Chicano/Latino	2.1354395318666763
22	2017	05	White	0.604163419314 6609
23	2017	05	Asian	0.7160300444150248
24	2017	05	Unknown	0.738335474466031
25	2017	05	International	0.8077799128051641
26	2017	05	African American	0.8498084505222766
27	2017	05	American Indian	1.0113543555965678
28	2017	05	Chicano/Latino	1.4412022594527392

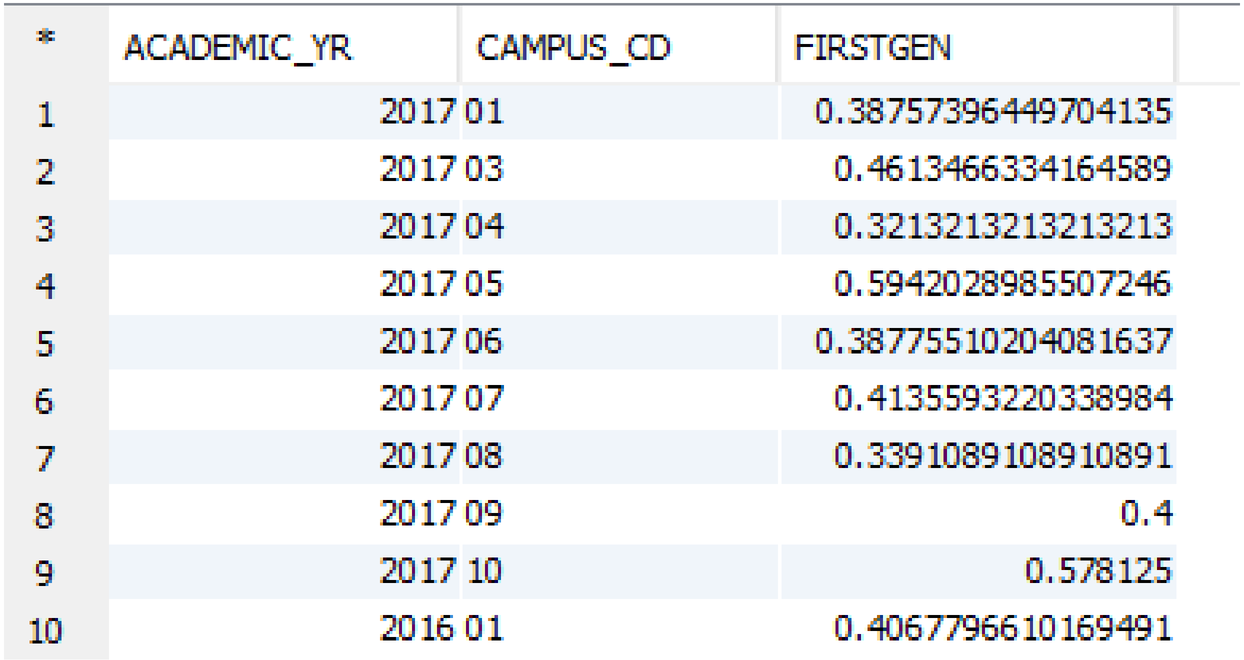
ucdug

Exercise

Don’t turn the page unless you want to see the answers

- Create a 2-year rolling average of the share of American Indian undergraduates who are first-generation, by campus. Thus, combine 2016 and 2017, 2015 and 2016, and so forth. Sort the results from most recent to oldest and then by campus code.
- Hint: Percent First Gen: **float**(**avg**(**CASE when** PARENT_EDUCATION_LVL='No College' **then** 1.0 **else** 0 **end**)
- Hint: Window range syntax: `ORDER BY ACADEMIC_YR asc RANGE BETWEEN 1 PRECEDING AND CURRENT ROW`

Expected result



*	ACADEMIC_YR	CAMPUS_CD	FIRSTGEN
1	2017	01	0.38757396449704135
2	2017	03	0.4613466334164589
3	2017	04	0.3213213213213213
4	2017	05	0.5942028985507246
5	2017	06	0.38775510204081637
6	2017	07	0.4135593220338984
7	2017	08	0.3391089108910891
8	2017	09	0.4
9	2017	10	0.578125
10	2016	01	0.4067796610169491

Sample Query

```sql
select distinct ACADEMIC_YR, CAMPUS_CD,
float(avg(CASE when PARENT_EDUCATION_LVL='No College' then
1.0 else 0 end) OVER (Partition by CAMPUS_CD order by
ACADEMIC_YR asc
RANGE BETWEEN 1 PRECEDING AND CURRENT ROW)) AS FirstGen
FROM IRAP_BI.ENROLLMENT_DM
WHERE STUD_LVL_UGR='Undergraduate' AND
ENR_UC_ETHN_6_CAT='American Indian'
ORDER BY ACADEMIC_YR desc, campus_cd
```

ucdug

OLAP SQL Functions

- Three types of functions:

1. Numbering: ROW_NUMBER
2. Ranking: LAG, LEAD, RANK, DENSE_RANK, PERCENT_RANK, CUME_DIST, NTILE
3. Aggregation: FIRST_VALUE, LAST_VALUE, RATIO_TO_REPORT, AVG, COUNT, MAX, MIN, RANGE, STDEV, SUM, VARIANCE

(This is for DB2, other databases may differ)