Problem Solving on Windows Functions

Relevel
by Unacademy

Instructions for the class

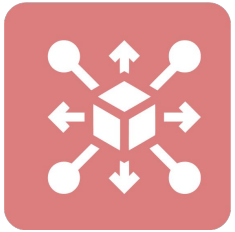
Instructions:

- We will use mode.com for this set of questions.

Caselet - 1

Caselet - 1

- **Tutorial.sat_scores**



Description:
This dataset is related to SAT scores. SAT is an exam used in USA to provide admission. SAT contains of three subjects - writing, verbal, and math. This dataset has following columns:

- School: The school in which student studied
- Teacher: Name of the teacher who taught the student
- Student_id: The id for each student(a unique identifier)
- Sat_writing: marks scored in writing
- Sat_verbal: marks scored in verbal
- Sat_maths: marks scored in math
- Hrs_studied: hours spent in studying
- Id: unique identifier for the dataset

Caselet - 1

Question-1:

Write a query to add column - avg_sat_writing. Each row in this column should include average marks in the writing section of the student per school.

Caselet - 1

Answer-1:

```
SELECT
  *,
  AVG(sat_writing)OVER(PARTITION BY school) AS avg_sat_writing
FROM
  Tutorial.sat_scores
```

Caselet - 1

Question-2:

In the above question, add an additional column - count_per_school. Each row of this column should include number of students per school

Caselet - 1

Answer-2:

```
SELECT
  *,
  AVG(sat_writing)OVER(PARTITION BY school) AS avg_sat_writing,
  COUNT(student_id)OVER(PARTITION BY school) AS count_per_school
FROM
  Tutorial.sat_scores
```

Caselet - 1

Question-3:

In the above question, add two additional columns - max_per_teacher and min_per_teacher. Each row of this column should include maximum and minimum marks in maths per teacher respectively.

Caselet - 1

Answer-3:

```
SELECT
  *,
  AVG(sat_writing)OVER(PARTITION BY school) AS avg_sat_writing,
  COUNT(student_id)OVER(PARTITION BY school) AS count_per_school,
  MAX(sat_math)OVER(PARTITION BY teacher) AS max_per_teacher,
  MIN(sat_math)OVER(PARTITION BY teacher) AS min_per_teacher
FROM
  tutorial.sat
```

Caselet - 1

Question-4:

For the dataset, write a query to add the two columns cum_hrs_studied and total_hrs_studied. Each row in cum_hrs_studied should display the cumulative sum of hours studied per school. Each row in the total_hrs_studied will display total hours studied per school. Order the data in the ascending order of student id

Caselet - 1

Answer-4

```
SELECT
  *,
  SUM(hrs_studied) OVER(PARTITION BY school ORDER BY student_id) AS cum_hrs_studied,
  SUM(hrs_studied) OVER(PARTITION BY school ORDER BY student_id ROWS BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS total_hrs_studied
FROM
  Tutorial.sat_scores
```

Caselet - 1

Question-5:

For the dataset, write a query to add column sub_hrs_studied and total_hrs_studied. Each row in sub_hrs_studied should display the sum of hrs_studied for a row above, a row below, and current row per school. Order the data in the ascending order of student id

Caselet - 1

Answer-5:

```
SELECT
  *,
  SUM(hrs_studied) OVER(PARTITION BY school ORDER BY student_id ROWS BETWEEN 1
PRECEDING AND 1 FOLLOWING) AS sub_hrs_studied
FROM
  Tutorial.sat_scores
```

Caselet - 1

Question-6:

Write a query to rank the students per school on the basis of scores in verbal. Use both rank and dense_rank function. Students with the highest marks should get rank 1.

Note: see if there is difference in ranking provided by both the functions.

Caselet - 1

Answer-6:

```
SELECT
  *,
 RANK() OVER(PARTITION BY school ORDER BY sat_verbal DESC) AS score_verbal_rank,
 DENSE_RANK() OVER(PARTITION BY school ORDER BY sat_verbal DESC) AS
score_verbal_dense_rank
FROM
  tutorial.sat_scores
```

Caselet - 1

Question-7:

Write a query to rank the students per school on the basis of scores in writing. Use both rank and dense_rank function. Student with the highest marks should get rank 1.

Note: see if there is difference in ranking provided by both the functions for teacher = ‘Spellman’

Caselet - 1

Answer-7:

```
SELECT
  *,
 RANK() OVER(PARTITION BY teacher ORDER BY sat_writing DESC) AS score_writing_rank,
 DENSE_RANK() OVER(PARTITION BY teacher ORDER BY sat_writing DESC) AS
score_writing_dense_rank
FROM
  Tutorial.sat_scores
```

Caselet - 1

Question-8:

Write a query to find the top 5 students per teacher who spent maximum hours studying.

Caselet - 1

Answer-8:

```
SELECT
  school,
  student_id
FROM
(
SELECT
  *,
 ROW_NUMBER()OVER(PARTITION BY teacher ORDER BY hrs_studied DESC) AS ranknum
FROM
  tutorial.sat_scores
) a
WHERE
  ranknum <6
```

Caselet - 1

Question-9:

Write a query to find the worst 5 students per school who got minimum marks in sat_math

Caselet - 1

Answer-9:

```
SELECT
  school,
  student_id
FROM
(
SELECT
  *,
 ROW_NUMBER()OVER(PARTITION BY school ORDER BY sat_math ) AS ranknum
FROM
  tutorial.sat_scores
) a
WHERE
  ranknum <6
```

Caselet - 1

Question-10:

Write a query to divide the dataset into quartile on the basis of marks in sat_verbal.

Caselet - 1

Answer-10:

```
SELECT
  *,
 NTILE(4)OVER( ORDER BY sat_verbal ) AS quartile
FROM
  tutorial.sat_scores
```

Caselet - 1

Question-11:

For ‘Petersville HS’ school, write a query to arrange the students in the ascending order of hours studied. Also, add a column to find the difference in hours studied from the student above(in the row). Exclude the cases where hrs_studied is null.

Caselet - 1

Answer-11:

```
SELECT
  *,
 hrs_studied - LAG(hrs_studied)OVER(ORDER BY hrs_studied) AS diff_hrs
FROM
  tutorial.sat_scores
WHERE
  school ='Petersville HS'
  AND hrs_studied IS NOT NULL
```

Caselet - 1

Question-12:

For ‘Washington HS’ school, write a query to arrange the students in the descending order of sat_math. Also, add a column to find the difference in sat_math from the student below(in the row).

Caselet - 1

Answer-12:

```
SELECT
  *,
 sat_math - LEAD(sat_math)OVER(ORDER BY sat_math DESC) AS diff_marks
FROM
  tutorial.sat_scores
WHERE
  school ='Washington HS'
```

Caselet - 1

Question-13:

Write a query to return 4 columns - student_id, school, sat_writing, difference in sat_writing and average marks scored in sat_writing in the school.

Caselet - 1

Answer-13:

```
SELECT
  student_id,
  school,
  sat_writing,
  sat_writing - AVG(sat_writing)OVER(PARTITION BY school) AS diff_avg
FROM
  tutorial.sat_scores
```

Caselet - 1

Question-14:

Write a query to return 4 columns - student_id, teacher, sat_verbal, difference in sat_verbal and minimum marks scored in sat_verbal per teacher.

Caselet - 1

Answer-14:

```
SELECT
  student_id,
  teacher,
  sat_verbal,
  sat_verbal - MIN(sat_verbal)OVER(PARTITION BY teacher) AS diff_min
FROM
  Tutorial.sat_scores
```

Caselet - 1

Question-15:

Write a query to return the student_id and school who are in bottom 20 in each of sat_verbal, sat_writing, and sat_math for their school.

Caselet - 1

Answer-15:

```
WITH data AS (
  SELECT
    student_id,
    school,
    ROW_NUMBER()OVER(PARTITION BY school ORDER BY sat_verbal) AS rank_verbal,
    ROW_NUMBER()OVER(PARTITION BY school ORDER BY sat_math) AS rank_math,
    ROW_NUMBER()OVER(PARTITION BY school ORDER BY sat_writing) AS rank_writing
  FROM
    tutorial.sat_scores
)
SELECT
  student_id,
  school
FROM
  data
WHERE
  rank_verbal < 21 AND rank_writing < 21 AND rank_math < 21
```

Caselet - 1

Question-16:

Write a query to find the student_id for the highest mark and lowest mark per teacher for sat_writing.

Caselet - 1

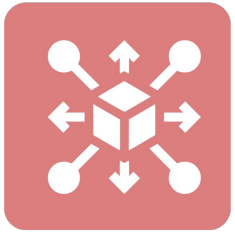
Answer-16:

```
SELECT DISTINCT
  teacher,
    FIRST_VALUE(student_id)OVER(PARTITION BY teacher ORDER BY sat_writing DESC ROWS
BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS max_marks_student,
    LAST_VALUE(student_id)OVER(PARTITION BY teacher ORDER BY sat_writing DESC ROWS
BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS min_marks_student
FROM
  tutorial.sat_scores
```

Caselet - 2

Caselet - 2

- **Tutorial.city_populations**



This dataset contains forecasts of the population of major cities of USA. The dataset has 4 columns:

- City: name of the city
- State: state name (in USA)
- Population_estimate_2012: forecast of population in 2012
- Id: the unique identifier of the dataset

Caselet - 2

Question-1:

Write a query to add an additional column - num_cities. Each row in the dataset should tell the number of cities in the dataset.

Caselet - 2

Answer-1:

```
SELECT *,
COUNT(city)OVER(PARTITION BY state) AS num_cities
FROM
  Tutorial.city_populations
```

Caselet - 2

Question-2:

Write a query to add an additional column - total_population. Each row in the dataset should tell the total population of the state.

Caselet - 2

Answer-2:

```
SELECT *,
SUM(population_estimate_2012)OVER(PARTITION BY state) AS
total_population
FROM
  tutorial.city_populations
```

Caselet - 2

Question-3:

Write a query to return the rows where population is more than the average population of the state

Caselet - 2

Answer-3:

```
WITH data AS (
  SELECT *,
  AVG(population_estimate_2012)OVER(PARTITION BY state) AS avg_population
  FROM
    tutorial.city_populations
)
SELECT *
FROM data
WHERE
  population_estimate_2012 > avg_population
```

Caselet - 2

Question-4:

Write a query to calculate the cumulative sum of population. Arrange the data in ascending order of the population.

Caselet - 2

Answer-4:

```
SELECT *,
SUM(population_estimate_2012)OVER(ORDER By population_estimate_2012) AS cum_population
FROM
  Tutorial.city_populations
```

Caselet - 2

Question-5:

Write a query to add a column rolling_avg. Each row in the dataset includes the average population for the two rows above and two rows below(including current row.

Caselet - 2

Answer-5:

```
SELECT *,
AVG(population_estimate_2012)OVER(ORDER By population_estimate_2012 ROWS BETWEEN 2
PRECEDING AND 2 FOLLOWING) AS rolling_avg
FROM
  Tutorial.city_populations
```

Caselet - 2

Question-6:

Write a query to rank the cities in California(CA) state in terms of population. City with the highest population is given rank 1. Use both rank and dense_rank function.

Caselet - 2

Answer-6:

```
SELECT *,
RANK()OVER(ORDER BY population_estimate_2012 DESC) AS population_rank,
DENSE_RANK()OVER(ORDER BY population_estimate_2012 DESC) AS population_dense_rank
FROM
  tutorial.city_populations
WHERE
  state ='CA'
```

Caselet - 2

Question-7:

Write a query to find the top 2 most populated cities per state.

Caselet - 2

Answer-7:

```
WITH data AS (
  SELECT
    *,
      ROW_NUMBER()OVER(PARTITION BY state ORDER BY population_estimate_2012 DESC) AS
population_dense_rank
  FROM
    tutorial.city_populations
)
SELECT
  city,
  state,
  population_dense_rank
FROM
  data
WHERE
  population_dense_rank < 3
```

Caselet - 2

Question-8:

Write a query to add a column - perc_pop. Each row in this column should represent the percentage of population a city contributes in that state.

Caselet - 2

Answer-8:

```
SELECT
  *,
  100.0*population_estimate_2012/SUM(population_estimate_2012)OVER(PARTITION BY state) AS
perc_pop
FROM
  Tutorial.city_populations
```

Caselet - 2

Question-9:

Write a query to find the cities which lie in the top 10 decile in terms of population

Caselet - 2

Answer-9:

```
SELECT
  *,
  NTILE(10)OVER(ORDER BY population_estimate_2012 DESC) AS percentile
FROM
  Tutorial.city_populations
```

Caselet - 2

Question-10:

Write a query to arrange the cities in the descending order of population and add a column calculating difference in population from 2 rows below (in the dataset).

Caselet - 2

Answer-10:

```
SELECT
  *,
  population_estimate_2012 - LEAD(population_estimate_2012)OVER(ORDER BY
population_estimate_2012 DESC) AS diff_pop
FROM
  Tutorial.city_populations
```

Caselet - 2

Question-11:

Write a query to return the state, first city and last city (in terms of id number) in the state.

Caselet - 2

Answer-11:

```
SELECT DISTINCT
  state,
 FIRST_VALUE(city) OVER(PARTITION BY state) AS first_city,
 LAST_VALUE(city) OVER(PARTITION BY state) AS last_city
FROM
  tutorial.city_populations
ORDER BY
  state
```