Practice – part 2: Complexe SELECT

Given a designed database that contains the following tables. The description in detail is found in Database description_Final_edudb_v2.doc file.

- student(**student_id**, first_name, last_name, dob, gender, address, note, *clazz_id*)
- subject(**subject_id**, name, credit, percentage_final_exam)
- lecturer(**lecturer_id**, first_name, last_name, dob, gender, address, email)
- teaching(***subject_id, lecturer_id***)
- clazz(**clazz_id**, name, *lecturer_id*, *monitor_id*)
- enrollment(***student_id, subject_id, semester,*** midterm_score, final_score)

Bài làm của sv: Các lệnh cần được lưu trong 1 file cho mỗi sinh viên, tên file: tensv_mssv-SQLPart2.sql. Ví dụ: oanhnt_20202201_SQLPart2.sql

11. Students aged 25 and above. Given information: student name, age
12. Students were born in June 1999.
13. Display class name and number of students corresponding in each class. Sort the result in descending order by the number of students.
14. Display the lowest, highest and average scores on the mid-term test of "Mạng máy tính" in semester '20172'.
15. Give number of subjects that each lecturer can teach. List must contain: lecturer id, lecturer's fullname, number of subjects.
16. List of subjects which have at least 2 lecturers in charge.
17. List of subjects which have less than 2 lecturers in charge.
18. List of students who obtained the highest score in subject whose id is 'IT3080', in the semester '20172'.

Functions

- *Aggregate functions* operate against a collection of values and return a single summarizing value.
- *Scalar functions* return a single value based on scalar input arguments

https://www.postgresql.org/docs/13/functions.html

Scalar functions

- *Scalar functions* return a single value based on scalar input arguments
- Can be used in any clause
- Example:

upper('tom') → TOM

lower('TOM') → tom

substring('Thomas' for 2) → Th

https://www.postgresql.org/docs/13/functions.html

Scalar functions

- *Scalar functions* return a single value based on scalar input arguments
- Example:

  current_date → 2021-04-09

  extract ('year' from current_date) → 2021

  age(current_date) → 00:00:00

select current_date, age(current_date), extract ('year' from current_date);

https://www.postgresql.org/docs/13/functions.html

Scalar functions

```
select current_date, age(current_date), extract ( 'year' from current_date);
```

Data Output | Explain | Messages | Notifications

	current_date date	age interval	date_part double precision
1	2021-04-09	00:00:00	2021

```
select subject_id, lower(subject_id), midterm_score, round(midterm_score)
from enrollment
where student_id = '20170002';
```

	subject_id character (6)	lower text	midterm_score double precision	round double precision
1	IT1110	it1110	7.5	8
2	IT3080	it3080	7.5	8
3	IT3090	it3090	[null]	[null]
4	IT3080	it3080	[null]	[null]
5	IT3090	it3090	[null]	[null]

https://www.postgresql.org/docs/13/functions.html

Scalar functions

```
select *, lower(subject_id)
from subject
where lower(subject_id) = 'it3090';
```

Data Output | Explain | Messages | Notifications

	subject_id [PK] character (6)	name character varying (30)	credit integer	percentage_final_exam integer	lower text
1	IT3090	Cơ sở dữ liệu	3	70	it3090

https://www.postgresql.org/docs/13/functions.html

Aggregate functions

- *Aggregate functions* compute a single result from a set of input values
- Some example: count(), avg(), max(), min (), sum(), …

MAX(): Computes the maximum of the non-null input values.

MIN(): Computes the minimum of the non-null input values.

AVG(): Computes the average (arithmetic mean) of all the non-null input values.

COUNT (*) : Computes the number of input rows. count ("any")

COUNT (attribute_name) : Computes the number of input rows in which the input value is not null.

COUNT(DISTINCT attribute_name) returns the number of unique non-null values in the attribute.

https://www.postgresql.org/docs/13/functions-aggregate.html

Aggregate functions

- Aggregate functions may be used in SELECT clause and HAVING clause
- An aggregation function can not be in WHERE clause, except it's in a sub-query.

```
update enrollment set final_score = null
where semester = '20172' and subject_id = 'IT3090'
            and student_id = '20170002';
```

```
select count(*), count(final_score), count(distinct final_score),
                max(final_score), min(final_score), avg(final_score)
from enrollment
where semester = '20172' and subject_id = 'IT3090';
```

https://www.postgresql.org/docs/13/functions-aggregate.html

Inner join vs. left join vs. right join

- *They are different*
- *They are not always interchangeable*

Exemple: Query N.16, N.17

https://www.postgresql.org/docs/13/functions.html

WITH clause

- WITH provides a way to write auxiliary statements for use in a larger query
- Each auxiliary statement in a WITH clause can be a SELECT, INSERT, UPDATE, or DELETE;
- All queries in the WITH list are computed ➔temporary tables that can be referenced in the FROM list. A WITH query that is **referenced more than once**in FROM is **computed only once**

WITH clause

- Exemple: query N.18

```
WITH tmp AS
   ( a sub-query )
SELECT  *
FROM tmp ;
```