18 October 2025

Exercise 2 : SQL Aggregate Functions & SQL Operators

Table : students

student_id	name	age	department
1	Alice	20	IT
2	Bob	22	HR
3	Charlie	21	IT
4	Diana	23	Finance
5	Eve	22	HR

1. List all distinct departments in the students table.

Expected columns:

* department

```
SELECT DISTINCT department
FROM students;
```

Return

department
IT
HR
IT
Finance
HR

2. Get the average age of students per department.

aggregate_function (column) As new_column

exp col : department, avg_age

```
SELECT department,
       AVG (age) As avg_age
FROM students
GROUP BY department;
```

IT dep Alice 20 + Charlie 21 = 20+21 = 41 / 2 = 20,5

HR dept
Bob 22 + Eve = 22 = 44/2 = 22

students

~~avg age~~ department	department avg age
~~20~~ IT	20,5
~~21~~ HR	22
Finance	11,5

IT dep
Bob 22, Charlie 21
Calcu: 22+21 = ~~43~~ / ~~2~~ = ~~21,5~~

Finance dep
Finance
Diana 23 / 2 = 11,5

3. Show departments (condition) with (more) than 1 student. (>)

Exp col: department, student_count.

~~HR dept~~
~~Diana 22, Eve 20~~
~~Cal 22+20~~
~~= 42~~
~~/ 2 = 21,5~~

~~SELECT COUNT (*)~~
SELECT department,
COUNT (*) As student_count
FROM students
GROUP BY department
HAVING COUNT (*) > 1 ;

(sorting statement)

(Allows to filter aggregated data)

department	student_count
IT	2
HR	2

4. Get all student whose age is between 21 and 23.

Exp col: student_id, name, age, department

SELECT student_id,
name,
age,
department,
FROM students
WHERE age ~~=21~~ BETWEEN 21
AND ~~OR~~ age BETWEEN 23;

Result: students

student_id	name	age	department
2	Bob	22	HR
4	Diana	23	Finance
5	Eve	22	HR

5. List all students in the IT or HR department who are older than 21.

Exp col: student_id, name, age, department

```
SELECT student_id,
       name,
       age,
       department,
FROM students
WHERE department IN('IT', 'HR')
      AND age > 21;
```

student_id	name	age	department
2	Bob	22	HR
5	Eve	22	HR

6. Show total credits per department, only for department with more than 5 total credits.

Exp col: department, total_credits

Table: courses

course_id	course_name	department	credits
101	SQL Basics	IT	3
102	Python	IT	4
103	Data Science	IT	4
104	Excel	Finance	2
105	Statistics	HR	3

```
SELECT department
       SUM (credits) As total_credits
FROM   courses
GROUPED BY department
HAVING total_credits > 5;
```

Return courses

department	total_credits
IT	11

7. List all courses that do not have 4 credits.
Exp col : course_id, courses_name, department, credits

```
SELECT ~~COUNT(~~ course_id,
                course_name,
                department,
                credits.
FROM courses
WHERE credits ~~=~~ NOT = 4;
```

courses

course_id	course_name	department	credits
101	SQL Basics	IT	3
104	Excel	Finance	2
105	Statistics	HR	3

8. Show the top 3 courses by credits in descending order.
Exp col : course_id, course_name, credits

```
SELECT  course_id,
        course_name,
FROM ~~cou~~ credits
FROM courses
ORDER BY credits DESC
LIMIT 3;
```

course_id	course_name	credits
103	Data Science	4
102	Python	4
101	SQL Basics	B 43

9. Get the maximum, minimum, and average grade across all enrollments.
Exp col : max_grade, min_grade, avg_grade
Table : enrollments

enrollment_id	student_id	course_id	grade
1	1	101	85
2	2	102	78
3	3	103	90
4	4	104	88
5	5	105	82

```
SELECT
    MAX (grade) As max_grade,
    MIN (grade) As min_grade,
    AVG (grade) As avg_grade,
FROM  enrollments;
```

90170
2 = 35

max_grade	min_grade	avg_grade
90	78	~~84.6~~ 8.4

10. Count how many enrollments exist per course.
Exp. col : course_id, enrollment_count

```
SELECT COUNT (*)  As enrollment_count
       course_id,
FROM   enrollments
GROUP BY course_id;
```

Return : enrollments

course_id	enrollment_count
101	1
102	1
103	1
104	1
105	1

11. Find total salary and total bonus per department.
Exp col : department, total_salary, total_bonus

Total : salaries

employee_id	name	department	salary	bonus
1	Tom	IT	60 000	5000
2	Jerry	HR	55 000	4000
3	Spike	Finance	70 000	6000
4	Tyke	IT	62 000	5500
5	Butch	HR	54 000	3500

```
SELECT  department,
        SUM(salary) As tot_salary
        SUM ( bonus) As tot_ bonus
FROM salaries
GROUP BY department;
```

Return Salaries

department	total_salary	total_bonus
IT	122 000	10 500
HR	109 000	7 500
Finance	70 000	6 000

12. Show departments where average salary is above 55,000.
Exp col : department, avg-salary

```
SELECT  department,
        AVG(salary) As avg-salary
FROM  salaries
WHERE  salary > 55 000
```

Salaries

department	avg-salary	
IT	61 000	122000 ÷ 2 = Avg
~~HR~~		
~~Fianct~~ ~~Finance~~		

13. List employees whose salary plus bonus is greater than 60,000.
Exp col : employee_id, name, salary, bonus, total_compensation

Table : projects

project_id	project_name	department	budget
1	AI App	IT	120 000
2	Payroll System	Finance	80 000
3	Dashboard	IT	150 000
4	Website	Marketing	60 000
5	HR Portal	HR	50 000

For question 14.

SELECT ~~project-id~~,
~~project-name~~,

SELECT employee-id,
name,
salary,
bonus,
FROM salaries
WHERE bonus AND salary > 60000

SELECT employee-id,
name,
salary,
bonus,
SUM(salary + bonus) As total-compensation
FROM salaries
WHERE (salary + bonus) > 60000

Salaries

employee-id	name	salary	bonus	total-compensation
1	Tom	60000	5000	65000
3	Spike	70000	6000	76000
4	Tyke	62000	5500	67500

14. Show total and average budget per department. Only include departments with average budget above 70 000.
Exp col : department, total_budget, avg-budget.

SELECT department,
SUM(budget) As total_budget,
AVG (budget) As avg-budget
FROM projects
GROUP BY department
HAVING AVG (budget) > 70 000;

department	total_budget	avg_budget
IT	270 000	135 000

15. List all projects with budgets between 50,000, excluding the Marketing department.

Exp col: project_id, project_name, department, budget

```
SELECT project_id,
       project_name,
       department,
       budget,
FROM projects
WHERE budget BETWEEN 50000 &
                 AND 120000
                 AND department NOT = 'Marketing';
```

Projects

project_id	project_name	department	budget
1	AI App	IT	120 000
2	Payroll System	Finance	80 000
3	Dashboard	IT	150 000
4			
5			