SQL SELECT STATEMENT

Aggregate Functions
GROUP BY & HAVING clauses

CT230
Database Systems

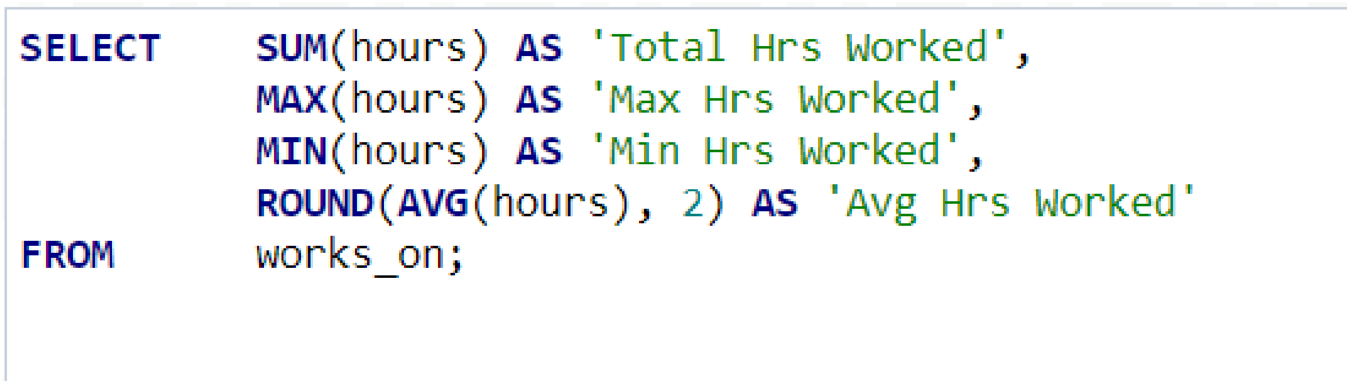
AGGREGATE FUNCTIONS

Aggregate functions are only supported (can only be used) in **SELECT** clause and **HAVING** clause, even if we would like to use them elsewhere! (e.g as part of a condition in where clause)

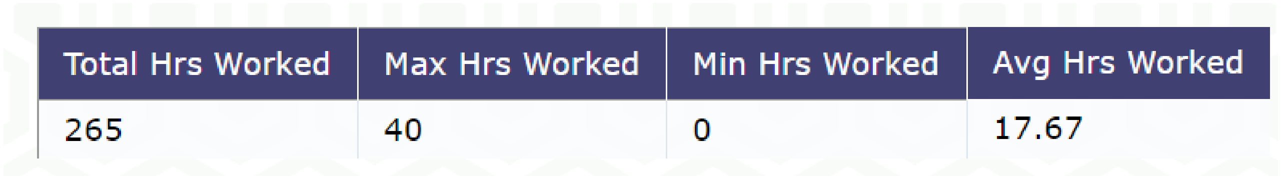
- Keywords **`SUM, AVG, MIN, MAX`** work as expected and can only be applied to **numeric** data
- Keyword **`COUNT`** can be used to count the number of tuples/values/rows specified in a query
- Can also use mathematical operations as part of an aggregate function on **numeric** data (e.g., *, +, -, /).

USING SUM, MAX, MIN, AVG

Example 23: Find the total number of hours worked on projects in the company, the maximum and minimum hours worked by an employee on a project and the average number of hours worked.



```
SELECT     SUM(hours) AS 'Total Hrs Worked',
           MAX(hours) AS 'Max Hrs Worked',
           MIN(hours) AS 'Min Hrs Worked',
           ROUND(AVG(hours), 2) AS 'Avg Hrs Worked'
FROM       works_on;
```



Total Hrs Worked	Max Hrs Worked	Min Hrs Worked	Avg Hrs Worked
265	40	0	17.67

DOES THIS MAKE SENSE?

```
SELECT    ssn, SUM(salary) AS answer
FROM      employee;
```

EXAMPLE 24 What is the output?

```
SELECT
    SUM(salary)/12
FROM
    employee;
```

To Do: Tidy up the output …

WORKING WITH `COUNT()`

- Very useful aggregate function
- Counts the **number of tuples/rows** in a result
- Can only be used in SELECT and HAVING clauses, as with all aggregate functions
- Similar to count() and counta() in Excel and other spreadsheets

EXAMPLE 25:
How *many* employees earn over 60000

** *Note:*

- Do not want the employee names
- Want to count how many there are
- Want a number returned…so we use count()

```
SELECT
   COUNT(*) AS 'num earning > 60k'
FROM
   employee
WHERE
   salary > 60000;
```

NOTE:

Whatever is in the output it is the tuples/rows which are counted …. therefore it is not necessary to specify the attribute name

```
SELECT
    COUNT(*) AS 'num earning > 60k'
FROM
    employee
WHERE
    salary > 60000;
```

MORE COUNT() EXAMPLES:

Example 26: Using a sub-query find how many employees work on project with name ‘ProductY’?

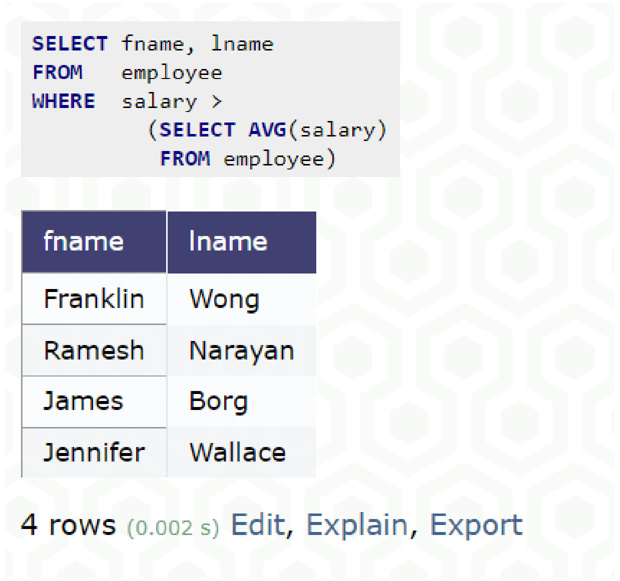
Example 27: Using a sub-query find how many children employee John Smith has?

Example 28: Find the yearly salary payments the company must make if everyone receives a 2% (.02) pay rise

Example 29: Find the number of employees working for the research department

USING A SUB-QUERY TO RETURN AN AGGREGATE VALUE

Example 30: Name the employees who earn greater than the average employee salary in the company



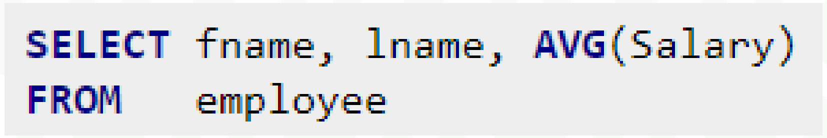
```
SELECT fname, lname
FROM   employee
WHERE  salary >
         (SELECT AVG(salary)
          FROM employee)
```

fname	lname
Franklin	Wong
Ramesh	Narayan
James	Borg
Jennifer	Wallace

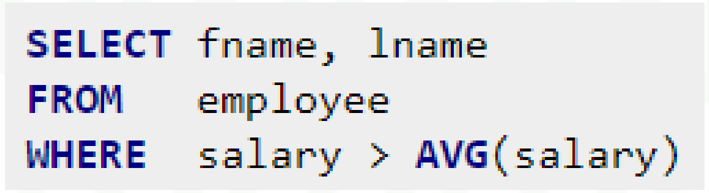
4 rows (0.002 s) Edit, Explain, Export

Only a subquery will work here

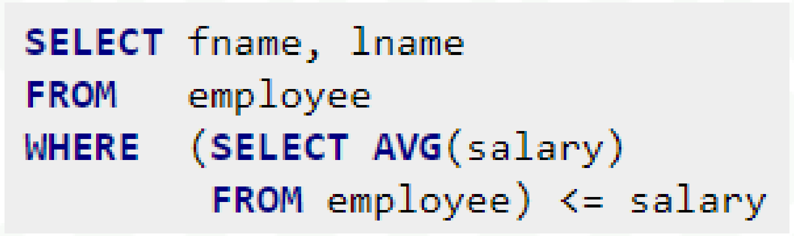
EXAMPLE 30 VARIATIONS
Will these work?



```
SELECT fname, lname, AVG(Salary)
FROM   employee
```



```
SELECT fname, lname
FROM   employee
WHERE  salary > AVG(salary)
```



```
SELECT fname, lname
FROM   employee
WHERE  (SELECT AVG(salary)
        FROM employee) <= salary
```

YOU TRY . . .

Example 31:

How many employees earn the minimum salary in the company?

GROUP BY HAVING

Recall:

```
SELECT [DISTINCT] <attribute list>
FROM <table list>
WHERE <condition>
GROUP BY <group attributes>
HAVING <group condition>
ORDER BY <attribute list>
```

GROUP BY

Syntax:

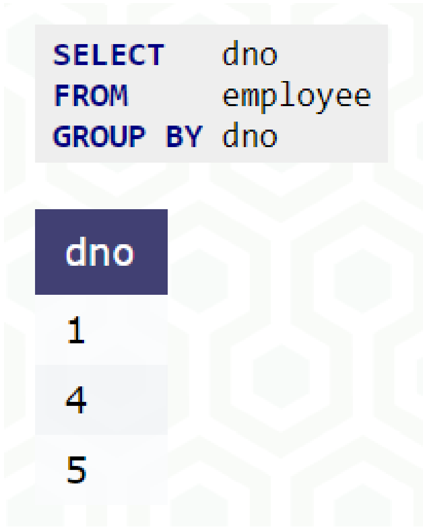
```
GROUP BY <group attributes>
```

- The GROUP BY clause allows the grouping (combining) of rows of a table together so that all occurrences within a specified group are collected together.
- Aggregate functions (min, max, avg, sum, count) can then be applied to the groups.

Example 32:

List the dno of each department

```
-- version 1
SELECT    dno
FROM      employee
GROUP BY  dno;

-- version 2
SELECT    DISTINCT dno
FROM      employee;
```



```
SELECT    dno
FROM      employee
GROUP BY  dno
```

dno
1
4
5

USING AGGREGATE FUNCTIONS WITH `GROUP BY` :

The GROUP BY clause specifies the group and the aggregate function is applied to the group.

- COUNT(*) can be used to *count* the number of rows (tuples) in the specified groups.
- AVG, SUM, MIN, MAX can be used to find average, sum, min and max of a *numerical value* in a specified group.

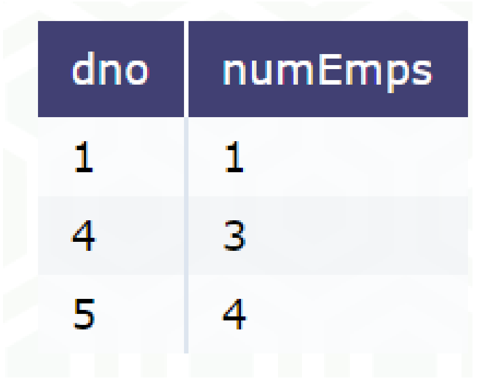
The aggregate function is not mentioned in the GROUP BY clause, but is specified in the SELECT clause.

* IMPORTANT *

You must GROUP BY ALL attributes mentioned in the SELECT clause *unless* they are involved in an aggregation.

EXAMPLE 33: List the department number and the number of employees in each department

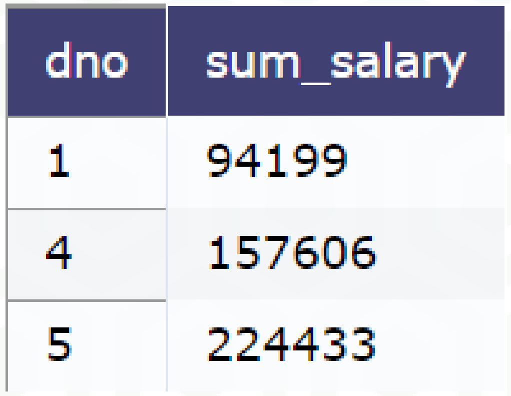
```
SELECT      dno, COUNT(*) AS numEmps
FROM        employee
GROUP BY    dno;
```



dno	numEmps
1	1
4	3
5	4

EXAMPLE 34: List the department number and the total salary in each department

```
SELECT   dno, SUM(salary) AS sum_salary
FROM     employee
GROUP BY dno;
```



dno	sum_salary
1	94199
4	157606
5	224433

You try … EXAMPLE 35: For each department, retrieve the department number, the number of employees in the department, and the average salary of the department

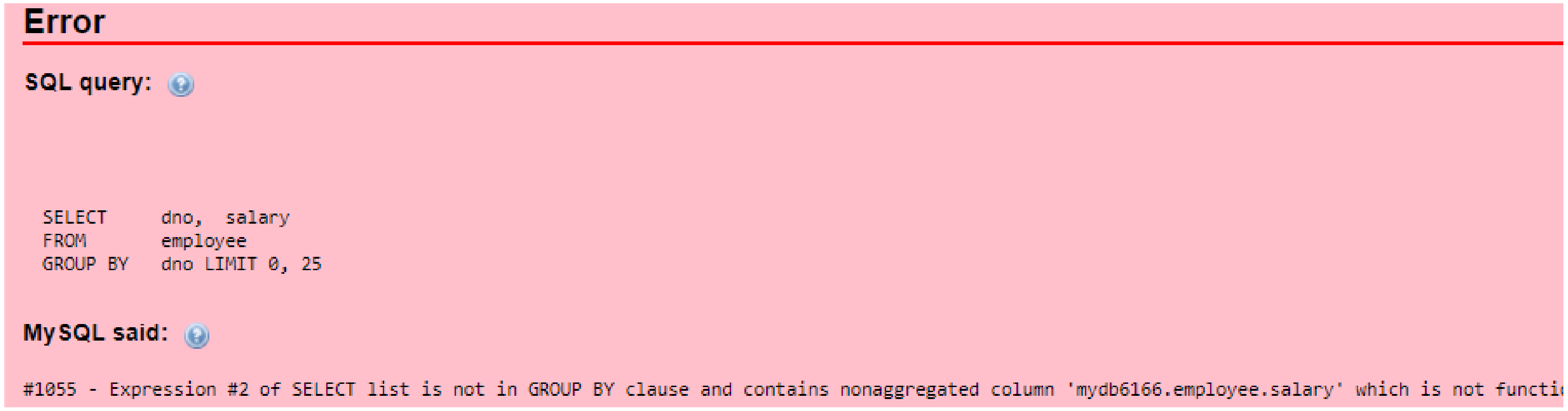
```
SELECT

FROM

GROUP BY
```

EXAMPLE 36:

List the number of dependents of each employee who has dependents

Why is this wrong?

```
SELECT      dno,  salary
FROM        employee
GROUP BY    dno;
```



Error

SQL query:

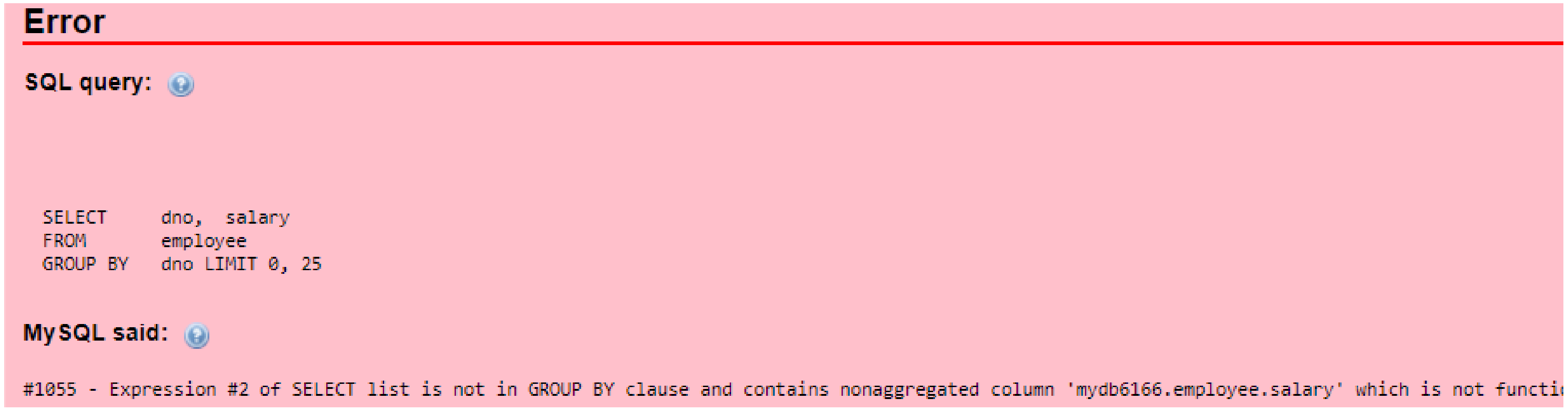
```
SELECT      dno,  salary
FROM        employee
GROUP BY    dno LIMIT 0, 25
```

MySQL said:

```
#1055 - Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'mydb6166.employee.salary' which is not functi
```

Recall:

- GROUP BY must contain all attributes in the SELECT clause that are not part of an aggregate function
- In the example, we cannot leave "salary" without a group



Error

SQL query:

```
SELECT     dno,  salary
FROM       employee
GROUP BY   dno LIMIT 0, 25
```

MySQL said:

```
#1055 - Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'mydb6166.employee.salary' which is not functi
```

HAVING

Syntax:

```
HAVING <group condition>
```

The HAVING clause is used in conjunction with `GROUP BY` and allows specification of conditions on groups.

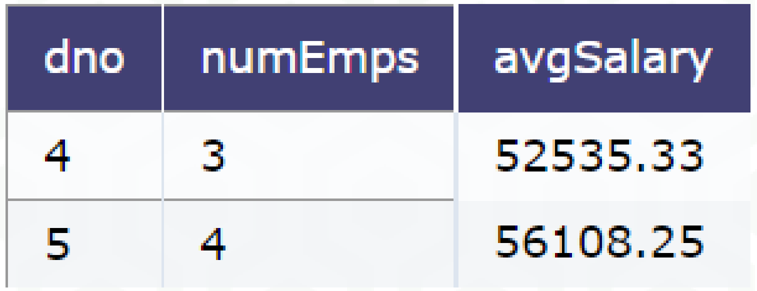
N.B. The column names used in the HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function, i.e., you cannot apply a HAVING condition to something that has not been calculated already.

Example 37: For each department that has more than 1 employee, retrieve the department number, the number of employees in the department and the average salary of the department.

```
SELECT     dno,
           COUNT(*) AS numEmps,
           AVG(salary) AS avgSalary
FROM       employee
GROUP BY   dno
HAVING     COUNT(*) > 1
```

Example 37: Tidying Output ...

```
SELECT      dno,
            COUNT(*) AS numEmps,
            CAST( AVG(salary) AS DECIMAL(10, 2)) AS avgSalary
FROM        employee
GROUP BY    dno
HAVING      COUNT(*) > 1
```



dno	numEmps	avgSalary
4	3	52535.33
5	4	56108.25

EXAMPLE 38: List the project number and the number of employees who work on the project for projects that have 2 or more employees

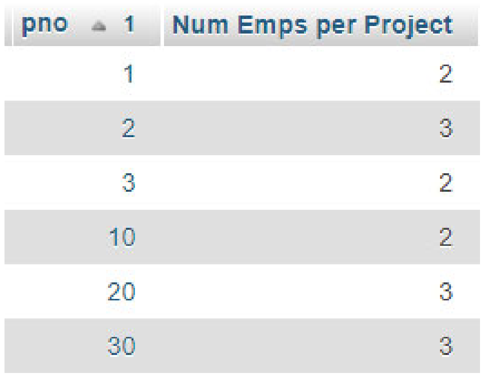
SELECT

FROM

GROUP BY

HAVING

ORDER BY



pno ▲ 1	Num Emps per Project
1	2
2	3
3	2
10	2
20	3
30	3

SUMMARY

Apart from Joins, have covered some of the most important aspects of SQL DDL and DML SELECT statements – with these you can build and query many databases.

Important to know:

- DDL `CREATE TABLE`
- DML `INSERT INTO`
- DML `SELECT`:
- Single table queries
- Multiple table queries with sub-queries *(To Do: Joins)*
- Aggregate functions
- Working with strings (LIKE, %, REGREP, etc.)
- Tidying Output (AS, CAST)