Session 5

Mohamed Emary

December 15, 2024

All the examples are on the ‘ITI‘ Database we have worked on before.

1 GROUP BY and HAVING

1.1 GROUP BY

In the `Instructor` table, to find the minimum salary between all the instructors, we can use:

```
SELECT
  MIN(Salary)
FROM
  Instructor;
```

But what if we want to find the minimum salary for each department? We can use the `GROUP BY` clause to group the rows based on the `Dept_name` and then apply the `MIN` function to each group.

```
SELECT
  MIN(Salary) "Min Salary Per Dept",
  Dept_Id
FROM
  Instructor
GROUP BY
  Dept_Id;
```

You may see a `NULL` value in the min salary column. This appears if there is one or more instructors in the department all having `NULL` salary. If there is at least one instructor with a salary, the `NULL` value will not appear (`NULL` is not a value and it only appears when there is no other value).

To avoid having `NULL` values in the result, we can use the `WHERE` clause to filter out the `NULL` values.

```
SELECT
  MIN(Salary) "Min Salary Per Dept",
  Dept_Id
```

```
FROM
  Instructor
WHERE
  Salary IS NOT NULL
GROUP BY
  Dept_Id;
```

The `GROUP BY` clause make the aggregation function (`MIN` in this case) apply to each group instead of the whole table.

Another way to apply the operation above is to use `PARTITION BY` in the `OVER` clause. This way, we can apply the aggregation function to each group without using the `GROUP BY` clause.

```
SELECT
  Dept_Id,
  MIN(Salary) OVER (
    PARTITION BY
      Dept_Id
  ) "Min Salary Per Dept"
FROM
  Instructor
WHERE
  Salary IS NOT NULL;
```

When using group by use it on a column that it's values are the same in multiple rows. Using it on a unique column like the primary key will be useless:

```
SELECT
  St_Id,
  COUNT(*)
FROM
  Student
GROUP BY
  St_Id;
```

You also can't group by `*`. For this to work you need to have at least two rows with the same values in all columns which shouldn't happen from the beginning since the primary key is unique. It's also not possible in code, you will get an error:

```
SELECT
  St_Id,
  COUNT(*)
FROM
  Student
GROUP BY
  * -- Error
```

To count the number of students in each department, the `COUNT` aggregate function would work on each group from the `GROUP BY` clause.

```
SELECT
  Dept_Id,
  COUNT(*) AS 'Number of Dep Students'
FROM
  Student
```

```sql
GROUP BY
  Dept_Id;

-- To ignore the NULL values in the Dept_Id column
SELECT
  Dept_Id,
  COUNT(*) AS 'Number of Dep Students'
FROM
  Student
WHERE
  Dept_Id IS NOT NULL
```

To count the number of students in the whole table:

```sql
SELECT
  -- Here the COUNT function will work on the whole table
  -- since there is no GROUP BY clause
  COUNT(*) AS 'Total Number of Students'
FROM
  Student;
```

Anything being selected next to the aggregate function and it's not an aggregate function should be in the `GROUP BY`:

```sql
SELECT
  St_Lname,
  COUNT(*)
FROM
  Student
GROUP BY
  St_Lname;

-- Another example
SELECT
  St_Fname,
  MAX(St_Age)
FROM
  Student
GROUP BY
  St_Fname;
```

Grouping by multiple columns have a similar idea to cross join:

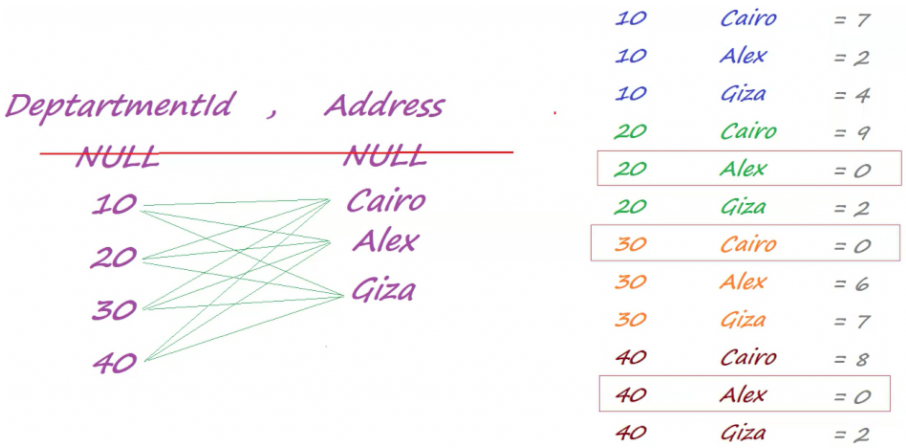

Figure 1: Grouping by multiple columns

```sql
SELECT
  Dept_Id,
  St_Address,
  COUNT(*) AS 'Number of Students'
FROM
  Student
WHERE
  Dept_Id IS NOT NULL
  AND St_Address IS NOT NULL
  -- Grouping here is done by address first (the second column)
  -- then by dept_id (the first column)
GROUP BY
  Dept_Id,
  St_Address;
```

1.2 HAVING

`SELECT` and `WHERE` work on table *record by record* while `COUNT` aggregate function works on the whole table, that is why you can't use `AND COUNT(*) > 2` in the `WHERE` clause in the statement below

If you want to get the number of students in each department that have more than 2 students:

```sql
-- This will not work
SELECT
  Dept_Id,
  COUNT(*) AS 'Number of Students'
FROM
  Student
WHERE
  Dept_Id IS NOT NULL
  AND COUNT(*) > 2
GROUP BY
  Dept_Id
```

To fix that you will need to use `HAVING` keyword, `HAVING` works on the groups created by the `GROUP BY` clause `HAVING` is mostly used with aggregate functions.

```sql
SELECT
  Dept_Id,
  COUNT(*) AS 'Number of Dep Students'
FROM
  Student
WHERE
  Dept_Id IS NOT NULL
GROUP BY
  Dept_Id
HAVING
  COUNT(*) > 2;
```

In general, we use `HAVING` without `GROUP BY` if we want to apply a condition on an aggregate function and we are not selecting that aggregate function in the `SELECT` clause **because we can't use aggregate functions in the `WHERE` clause**.

With `HAVING` we always have a condition with aggregation that condition works on the groups created by the `GROUP BY` clause or on the whole table if there is no `GROUP BY` clause and we are using an aggregate function in the `SELECT` clause.

Some general rules:

1. Aggregate functions like `COUNT` work on the whole table.
2. `WHERE` works on the table record by record.
3. `HAVING` works on the groups created by the `GROUP BY` clause.

Example of using `HAVING` without `GROUP BY`, we want to get the sum of all instructors salaries if there is more than 10 instructors in the table:

```sql
SELECT
  SUM(Salary) AS 'Total Salaries'
FROM
  Instructor
HAVING
  COUNT(*) > 10;
```

If we want to get the sum of salaries for each department, we can use one of the two statements below.

The two statement below is similar to each other but the second one uses join. The performance in the second one is worse than the first since we are applying operations on two tables

Generally this is not a good case for using join. Join is used when we want to get data from two tables

```sql
SELECT
  Dept_Id,
  SUM(Salary) AS 'SumOfSalaries'
FROM
  Instructor
WHERE
  Dept_Id IS NOT NULL
GROUP BY
  Dept_Id;

```

```sql
-- Using join
SELECT
  I.Dept_Id,
  SUM(I.Salary) AS 'SumOfSalaries'
FROM
  Instructor I,
  Department D
WHERE
  I.Dept_Id = D.Dept_Id
GROUP BY
  I.Dept_Id;
```

Here each department have a different name, and to show the department name here we still need to group by `Dept_Name` to make the query work:

```sql
SELECT
  I.Dept_Id,
  D.Dept_Name,
  SUM(I.Salary) AS 'SumOfSalaries'
FROM
  Instructor I,
  Department D
WHERE
  I.Dept_Id = D.Dept_Id
GROUP BY
  I.Dept_Id,
  D.Dept_Name;
```

To select the students who act as supervisors and the number of students they supervise, we had to group by both `Supr.St_Fname`, `Supr.St_Id` since we are selecting both of them in the `SELECT` clause If we are selecting only one of them we can group by only that column.

> ***NOTE:*** If you group by only the `St_Fname` column, and we have two supervisors with the same `St_Fname`, the query will group them together and show the total number of students they supervise together.

```sql
SELECT
  Supr.St_Fname 'Supervisor',
  Supr.St_Id 'Supervisor ID',
  COUNT(*) 'No of Students'
FROM
  Student Stud,
  Student Supr
WHERE
  Supr.St_Id = Stud.St_super
GROUP BY
  Supr.St_Fname,
  Supr.St_Id;
```