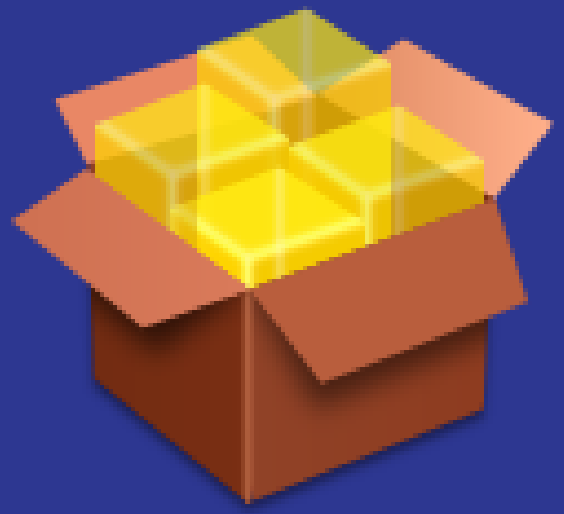
Data Aggregation

How to get data insights?

Table of Content



Grouping

Consolidating data based on criteria

Grouping (1)

- Grouping allows taking data into separate groups based on a common property

Employee	DepartmentName	Salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

Grouping (1)

Grouping column

Single row

Can be aggregated

Employee	DepartmentName	Salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

Grouping (2)

- With **GROUP BY** you can get each separate group and use an "aggregate" function over it (like Average, Min or Max):

```
   SELECT e.DepartmentID
     FROM Employees AS e
GROUP BY e.DepartmentID
```

Grouping Columns

- With **DISTINCT** you will get all unique values:

```
SELECT DISTINCT e.DepartmentID
  FROM Employees AS e
```

Unique Values

Problem: Departments Total Salaries

- Use "SoftUni" database to create a query which prints the total sum of salaries for each department.
  - Order them by DepartmentID (ascending).

Employee	DepartmentID	Salary
Adam	1	5,000
John	1	15,000
Jane	2	10,000
George	2	15,000
Lila	2	5,000
Fred	3	15,000

DepartmentID	TotalSalary
1	20,000
2	30,000
3	15,000

Solution: Departments Total Salaries

- After grouping every employee by it's department we can use aggregate function to calculate total amount of money per group.

Grouping Column

Column Alias

```
SELECT e.DepartmentID,
  SUM(e.Salary) AS TotalSalary
FROM Employees AS e
GROUP BY e.DepartmentID
ORDER BY e.DepartmentID
```

Table Alias

Grouping Column

GROUP BY with WHERE

- The WHERE clause can also be used with GROUP BY clause to restrict the rows for grouping.
- The rows that satisfy the search condition are considered for grouping.

Solution: Departments Total Salaries

- Create a query which prints the total sum of salaries for each department with salary of employee must be greater than 5000

Employee	DepartmentID	Salary
Adam	1	5,000
John	1	15,000
Jane	2	10,000
George	2	15,000
Lila	2	5,000
Fred	3	15,000

DepartmentID	TotalSalary
1	15,000
2	25,000
3	15,000

Solution: Departments Total Salaries

- After grouping every employee by it's department we can use aggregate function to calculate total amount of money per group.

Grouping Column

Column Alias

```
SELECT e.DepartmentID,
  SUM(e.Salary) AS TotalSalary
FROM Employees AS e
WHERE Salary > 5000
GROUP BY e.DepartmentID
ORDER BY e.DepartmentID
```

Table Alias

Condition

Grouping Column

Aggregate Functions

COUNT, SUM, MAX, MIN, AVG…

Aggregate Functions

- Operate over (non-empty) groups
- Perform data analysis on each one
  - **MIN**, **MAX**, **AVG**, **COUNT** etc.

```
SELECT e.DepartmentID,
 MIN(e.Salary) AS MinSalary
FROM Employees AS e
GROUP BY e.DepartmentID
```

	DepartmentID	MinSalary
1	1	32700.00
2	2	25000.00
3	3	23100.00
4	4	13500.00
5	5	12800.00
6	6	40900.00

- Aggregate functions usually ignore **NULL** values.

Aggregate Functions: COUNT

- **COUNT** - count the values in one or more grouped columns
  - Ignores `null` values

Employee	DepartmentName	Salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

DepartmentName	SalaryCount
Database Support	2
Application Support	3
Software Support	1

COUNT Syntax

- **COUNT**(*ColumnName*)

Grouping Column

New Column Alias

```
   SELECT e.DepartmentID,
          COUNT(e.Salary) AS SalaryCount
     FROM Employees AS e
GROUP BY e.DepartmentID
```

Grouping Columns

- Note: **COUNT** ignores any employee with **NULL** salary.

Aggregate Functions: SUM

- **SUM** - sums the values in a column.

Employee	DepartmentName	Salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

DepartmentName	TotalSalary
Database Support	20,000
Application Support	30,000
Software Support	15,000

SUM Syntax

- If any department has no salaries, it returns NULL.

Grouping Column

New Column Alias

```
SELECT e.DepartmentID,
       SUM(e.Salary) AS TotalSalary
  FROM Employees AS e
GROUP BY e.DepartmentID
```

Table Alias

Grouping Columns

Aggregate Functions: MAX

- **MAX** - takes the largest value in a column.

Employee	DepartmentName	Salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

DepartmentName	MaxSalary
Database Support	15,000
Application Support	15,000
Software Support	15,000

MAX Syntax

Grouping Column

New Column Alias

```
   SELECT e.DepartmentID,
          MAX(e.Salary) AS MaxSalary
     FROM Employees AS e
GROUP BY e.DepartmentID
```

Table Alias

Grouping Columns

Aggregate Functions: MIN

- `MIN` takes the smallest value in a column.

Employee	DepartmentName	Salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

DepartmentName	MinSalary
Database Support	5,000
Application Support	5,000
Software Support	15,000

MIN Syntax

Grouping Column

New Column Alias

```
   SELECT e.DepartmentID,
          MIN(e.Salary) AS MinSalary
     FROM Employees AS e
GROUP BY e.DepartmentID
```

Table Alias

Grouping Columns

Aggregate Functions: AVG

- **AVG** calculates the average value in a column.

Employee	DepartmentName	Salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

DepartmentName	AvgSalary
Database Support	10,000
Application Support	10,000
Software Support	15,000

AVG Syntax

Grouping Column

New Column Alias

```
   SELECT e.DepartmentID,
          AVG(e.Salary) AS AvgSalary
     FROM Employees AS e
GROUP BY e.DepartmentID
```

Table Alias

Grouping Columns

Having

Using predicates while grouping

Having Clause

- The **HAVING** clause is used to filter data based on aggregate values
  - We cannot use it without grouping first
- Aggregate functions (**MIN**, **MAX**, **SUM** etc.) are executed only once
  - Unlike **HAVING**, **WHERE** filters rows before aggregation

HAVING Clause: Example

- Filter departments having total salary more than or equal to 15,000

Aggregated value

Employee	DepartmentName	Salary	TotalSalary
Adam	Database Support	5,000	20,000
John	Database Support	15,000	
Jane	Application Support	1,000	11,000
George	Application Support	5,000	
Lila	Application Support	5,000	
Fred	Software Support	15,000	15,000

DepartmentName	TotalSalary
Database Support	20,000
Software Support	15,000

HAVING Syntax

Grouping Column

Aggregate Function

Column Alias

```
   SELECT e.DepartmentID,
          SUM(e.Salary) AS TotalSalary
     FROM Employees AS e
GROUP BY e.DepartmentID
   HAVING SUM(e.Salary) < 250000
```

Grouping Columns

Having Predicate

Sales Table with daily sales data

SalesId	SalesDate
1	2011-06-12 01:12:05.490
2	2011-06-13 01:12:05.630
3	2011-06-13 01:12:05.777
4	2011-06-14 01:12:05.750
5	2011-06-15 01:12:05.690
6	2011-06-17 01:12:05.777
7	2011-06-18 01:12:05.773
...	
...	
...	

Quarterly Sales data

Year	Quarter	Sales Count
2011	2	15
2011	3	100
2011	4	106
2012	1	83
2012	2	83
2012	3	93
2012	4	84
2013	1	85
2013	2	111
2013	3	77
2013	4	100
2014	1	63

Pivot Tables

Pivot Tables

- Summarizes data from another table
- Applies an aggregate operation
  - (sorting, averaging, summing, etc…)
- Typically includes grouping of the data.

Pivot Tables

- Syntax

```
SELECT <non-pivoted column>,
    [first pivoted column] AS <column name>,
    [second pivoted column] AS <column name>,
    ...
    [last pivoted column] AS <column name>
FROM
    (<SELECT query that produces the data>)
     AS <alias for the source query>
PIVOT
(
    <aggregation function>(<column being aggregated>)
FOR
[<column that contains the values that will become column headers>]
    IN ( [first pivoted column], [second pivoted column],
    ... [last pivoted column])
) AS <alias for the pivot table>
[<ORDER BY clause>];
```

Pivot Tables - Example

```
SELECT SUBID,[SV01] AS STUDENT1,[SV02] AS STUDENT2,
             [SV03] AS STUDENT3,[SV04] AS STUDENT4
FROM (SELECT ROLLNO, SUBID, Marks FROM EXAMMARK) P
PIVOT
(
      SUM(Marks) FOR ROLLNO
      IN([SV01],[SV02],[SV03],[SV04])
) PVT
```

Pivoted column

Column become header

Alias for the source query

Alias for the pivot table

aggregation function

Summary

1. Grouping by Shared Properties
2. Aggregate Functions
3. Having Clause

```
SELECT
  SUM(e.Salary) AS 'TotalSalary'
FROM Employees AS e
GROUP BY e.DepartmentID
HAVING SUM(e.Salary) < 250000
```

4. Pivot Tables