SQL | GROUP BY

The GROUP BY Statement in SQL is used to arrange identical data into groups with the help of some functions. i.e. if a particular column has the same values in different rows then it will arrange these rows in a group.

Features

- GROUP BY clause is used with the SELECT statement.
- In the query, the GROUP BY clause is placed after the WHERE clause.
- In the query, the GROUP BY clause is placed before the ORDER BY clause if used.
- In the query, the Group BY clause is placed before the Having clause.
- Place condition in the having clause.

Syntax:

```
SELECT column1, function_name(column2)
FROM table_name
```



```
WHERE condition
GROUP BY column1, column2
ORDER BY column1, column2;
```

Explanation:

1. **function_name**: Name of the function used for example, SUM() , AVG().

2. **table_name**: Name of the table.
3. **condition**: Condition used.

Let's assume that we have two tables Employee and Student Sample Table is as follows after adding two tables we will do some specific operations to learn about GROUP BY.

Employee Table:

```
CREATE TABLE emp (
  emp_no INT PRIMARY KEY,
  name VARCHAR(50),
  sal DECIMAL(10,2),
  age INT
);
```

Insert some random data into a table and then we will perform some operations in GROUP BY.

Query:

```
INSERT INTO emp (emp_no, name, sal, age) VALUES
(1, 'Aarav', 50000.00, 25),
(2, 'Aditi', 60000.50, 30),
(3, 'Amit', 75000.75, 35),
(4, 'Anjali', 45000.25, 28),
(5, 'Chetan', 80000.00, 32),
(6, 'Divya', 65000.00, 27),
(7, 'Gaurav', 55000.50, 29),
(8, 'Isha', 72000.75, 31),
(9, 'Kavita', 48000.25, 26),
(10, 'Mohan', 83000.00, 33);
```

Output:

emp_no	name	sal	age
1	Aman	50000	25
2	Shubham	60000.5	30
3	Aditya	75000.75	35
4	Navin	45000.25	28
5	Chetan	80000	32
6	Divya	65000	27
7	Gaurav	55000.5	29
8	Isha	72000.75	31
9	Kavita	48000.25	26
10	Mohan	83000	33

Student Table:

Query:

```
CREATE TABLE student (
  name VARCHAR(50),
  year INT,
  subject VARCHAR(50)
);
INSERT INTO student (name, year, subject) VALUES
('Alice', 1, 'Mathematics'),
('Bob', 2, 'English'),
('Charlie', 3, 'Science'),
('David', 1, 'History'),
('Emily', 2, 'Art'),
('Frank', 3, 'Computer Science');
```

Output:

name	year	subject
Alice	1	Mathematics
Bob	2	English
Charlie	3	Science
David	1	History
Emily	2	Art
Frank	3	Computer Science

Group By single column

Group By single column means, placing all the rows with the same value of only that particular column in one group. Consider the query as shown below:

Query:

```
SELECT NAME, SUM(SALARY) FROM emp
GROUP BY NAME;
```

The above query will produce the below output:

name	SUM(sal)
Aditya	75000.75
Aman	50000
Chetan	80000
Divya	65000
Gaurav	55000.5
Isha	72000.75
Kavita	48000.25
Mohan	83000
Navin	45000.25

As you can see in the above output, the rows with duplicate NAMEs are grouped under the same NAME and their corresponding SALARY is the sum of the SALARY of duplicate rows. The SUM() function of SQL is used here to calculate the sum.

Group By Multiple Columns

Group by multiple columns is say, for example, **GROUP BY column1, column2**. This means placing all the rows with the same values of columns **column 1** and **column 2** in one group. Consider the below query:

Query:

```
SELECT SUBJECT, YEAR, Count(*)
FROM Student
GROUP BY SUBJECT, YEAR;
```

Output:

subject	year	Count(*)
Art	2	1
Computer Science	3	1
English	2	1
History	1	1
Mathematics	1	1
Science	3	1

Output: As you can see in the above output the students with both the same SUBJECT and YEAR are placed in the same group. And those whose only SUBJECT is the same but not YEAR belong to different groups. So here we have grouped the table according to two columns or more than one column.

HAVING Clause in GROUP BY Clause

We know that the WHERE clause is used to place conditions on columns but what if we want to place conditions on groups? This is where the HAVING clause comes into use. We can use the HAVING clause to place conditions to decide which group will be part of the final result set.

Also, we can not use aggregate functions like SUM(), COUNT(), etc. with the WHERE clause. So we have to use the HAVING clause if we want to use any of these functions in the conditions.

Syntax:

SELECT column1, function_name(column2)

FROM table_name

WHERE condition

GROUP BY column1, column2

HAVING condition

ORDER BY column1, column2;

Explanation:

1. **function_name**: Name of the function used for example, SUM() , AVG().
2. **table_name**: Name of the table.
3. **condition**: Condition used.

Example:

```
SELECT NAME, SUM(sal) FROM Emp
GROUP BY name
HAVING SUM(sal)>3000;
```

Output:

name	SUM(sal)
Aditya	75000.75
Aman	50000
Chetan	80000
Divya	65000
Gaurav	55000.5
Isha	72000.75
Kavita	48000.25
Mohan	83000
Navin	45000.25

As you can see in the above output only one group out of the three groups appears in the result set as it is the only group where sum of SALARY is greater than 3000. So we have used the HAVING clause here to place this condition as the condition is required to be placed on groups not columns.

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 06 May, 2023

166

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. Difference between order by and group by clause in SQL
5. Group by clause in MS SQL Server
6. How to Group and Aggregate Data Using SQL?
7. SQL - Using GROUP BY to COUNT the Number of Rows For Each Unique Entry in a Column
8. SQL - count() with Group By clause
9. How to Select Group of Rows that Match All Items on a List in SQL Server?
10. How to Select the First Row of Each GROUP BY in SQL?

Previous Next

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

SQL – ORDER BY

The ORDER BY statement in SQL is used to sort the fetched data in either ascending or descending according to one or more columns.

- By default ORDER BY sorts the data in ascending order.
- We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.

Sort According To One Column

To sort in ascending or descending order we can use the keywords ASC or DESC respectively.

Syntax:

*SELECT * FROM table_name ORDER BY column_name ASC | DESC*

//Where

***table_name**: name of the table.*

***column_name**: name of the column according to which the data*

is needed to be arranged.

***ASC**: to sort the data in ascending order.*

***DESC**: to sort the data in descending order.*

***|** : use either ASC or DESC to sort in ascending or descending order//*

Sort According To Multiple Columns

To sort in ascending or descending order we can use the keywords ASC or DESC respectively. To sort according to multiple columns, separate the names of columns by the (,) operator.

Syntax:

*SELECT * FROM table_name ORDER BY column1 ASC|DESC , column2 ASC|DESC*

Query:

```
CREATE TABLE students (
  roll_no INT NOT NULL,
  age INT NOT NULL,
  name VARCHAR(50) NOT NULL,
  address VARCHAR(100) NOT NULL,
  phone VARCHAR(20) NOT NULL,
  PRIMARY KEY (roll_no)
);
INSERT INTO students (roll_no, age, name, address, phone)
VALUES
  (1, 18, 'Shubham Thakur', '123 Main St, Mumbai', '9876543210'),
  (2, 19, 'Aman Chopra', '456 Park Ave, Delhi', '9876543211'),
  (3, 20, 'Naveen Tulasi', '789 Broadway, Ahmedabad', '9876543212'),
  (4, 21, 'Aditya arpan', '246 5th Ave, Kolkata', '9876543213'),
  (5, 22, 'Nishant Jain', '369 3rd St, Bengaluru', '9876543214');
```

Output:

roll_no	age	name	address	phone
1	18	Shubham Thakur	123 Main St, Mumbai	9876543210
2	19	Aman Chopra	456 Park Ave, Delhi	9876543211
3	20	Naveen Tulasi	789 Broadway, Ahmedabad	9876543212
4	21	Aditya arpan	246 5th Ave, Kolkata	9876543213
5	22	Nishant Jain	369 3rd St, Bengaluru	9876543214

Now consider the above database table and find the results of different queries.

Sort According To a Single Column

In this example, we will fetch all data from the table Student and sort the result in descending order according to the column ROLL_NO.

Query:

```
SELECT * FROM students ORDER BY ROLL_NO DESC;
```

Output:

roll_no	age	name	address	phone
5	22	Nishant Jain	369 3rd St, Bengaluru	9876543214
4	21	Aditya arpan	246 5th Ave, Kolkata	9876543213
3	20	Naveen Tulasi	789 Broadway, Ahmedabad	9876543212
2	19	Aman Chopra	456 Park Ave, Delhi	9876543211
1	18	Shubham Thakur	123 Main St, Mumbai	9876543210

In the above example, if we want to sort in ascending order we have to use ASC in place of DESC.

Sort According To Multiple Columns

In this example, we will fetch all data from the table Student and then sort the result in ascending order first according to the column Age. and then in descending order according to the column ROLL_NO.

Query:

```
SELECT * FROM students ORDER BY Age ASC , ROLL_NO DESC;
```

Output:

roll_no	age	name	address	phone
1	18	Shubham Thakur	123 Main St, Mumbai	9876543210
2	19	Aman Chopra	456 Park Ave, Delhi	9876543211
3	20	Naveen Tulasi	789 Broadway, Ahmedabad	9876543212
4	21	Aditya arpan	246 5th Ave, Kolkata	9876543213
5	22	Nishant Jain	369 3rd St, Bengaluru	9876543214

In the above output, we can see that first the result is sorted in ascending order according to Age. There are multiple rows of having the same Age. Now, sorting further this result-set according to ROLL_NO will sort the rows with the same Age according to ROLL_NO in descending order.

Note:

ASC is the default value for the ORDER BY clause. So, if we don't specify anything after the column name in the ORDER BY clause, the output will be sorted in ascending order by default.

Take another example of the following query that will give similar output as the above:

Query:

```
SELECT * FROM students ORDER BY Age , ROLL_NO DESC;
```

Output:

roll_no	age	name	address	phone
1	18	Shubham Thakur	123 Main St, Mumbai	9876543210
2	19	Aman Chopra	456 Park Ave, Delhi	9876543211
3	20	Naveen Tulasi	789 Broadway, Ahmedabad	9876543212
4	21	Aditya arpan	246 5th Ave, Kolkata	9876543213
5	22	Nishant Jain	369 3rd St, Bengaluru	9876543214

Sorting By Column Number (instead of name)

An integer that identifies the number of the column in the SelectItems in the underlying query of the SELECT statement. Column number must be greater than 0 and not greater than the number of columns in the result table. In other words, if we want to order by a column, that column must be specified in the SELECT list.

The rule checks for ORDER BY clauses that reference select list columns using the column number instead of the column name. The column numbers in the ORDER BY clause impair the readability of the SQL statement. Further, changing the order of columns in the SELECT list has no impact on the ORDER BY when the columns are referred to by names instead of numbers.

Syntax:

Order by Column_Number asc/desc

Here we take an example to sort a database table according to column 1 i.e Roll_Number. For this a query will be:

Query:

```
CREATE TABLE studentinfo
( Roll_no INT,
NAME VARCHAR(25),
Address VARCHAR(20),
CONTACTNO BIGINT NOT NULL,
Age INT );

INSERT INTO studentinfo
VALUES (7,'ROHIT','GHAZIABAD',9193458625,18),
(4,'DEEP','RAMNAGAR',9193458546,18),
(1,'HARSH','DELHI',9193342625,18),
```

```
(8,'NIRAJ','ALIPUR',9193678625,19),
(5,'SAPTARHI','KOLKATA',9193789625,19),
(2,'PRATIK','BIHAR',9193457825,19),
(6,'DHANRAJ','BARABAJAR',9193358625,20),
(3,'RIYANKA','SILIGURI',9193218625,20);

SELECT Name, Address
FROM studentinfo
ORDER BY 1
```

Output:

NAME	Address
DEEP	RAMNAGAR
DHANRAJ	BARABAJAR
HARSH	DELHI
NIRAJ	ALIPUR
PRATIK	BIHAR
RIYANKA	SILIGURI
ROHIT	GHAZIABAD
SAPTARHI	KOLKATA

Last Updated : 03 May, 2023

47

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. SQL | Procedures in PL/SQL
5. SQL | Difference between functions and stored procedures in PL/SQL
6. SQL SERVER – Input and Output Parameter For Dynamic SQL
7. Difference between T-SQL and PL-SQL
8. Difference between SQL and T-SQL
9. SQL Server | Convert tables in T-SQL into XML

SQL HAVING Clause with Examples

xishaapatel

The HAVING clause was introduced in SQL to allow the filtering of query results based on aggregate functions and groupings, which cannot be achieved using the WHERE clause that is used to filter individual rows.

In simpler terms MSSQL, the HAVING clause is used to apply a filter on the result of GROUP BY based on the specified condition. The conditions are Boolean type i.e. *use of logical operators (AND, OR).* This clause was included in SQL as the WHERE keyword failed when we use it with aggregate expressions. Having is a very generally used clause in SQL. Similar to WHERE it helps to apply conditions, but HAVING works with groups. If you wish to filter a group, the HAVING clause comes into action.

Some important points:

- Having clause is used to filter data according to the conditions provided.
- Having a clause is generally used in reports of large data.
- Having clause is only used with the SELECT clause.
- The expression in the syntax can only have constants.
- In the query, ORDER BY is to be placed after the HAVING clause, if any.
- HAVING Clause is implemented in column operation.
- Having clause is generally used after GROUP BY.

Syntax:


```
SELECT col_1, function_name(col_2)
```

FROM tablename

WHERE condition

GROUP BY column1, column2

HAVING Condition

ORDER BY column1, column2;

Here, the function_name is the name of the function used, for example, SUM(), and AVG().

Example 1:

Here first we create a database name as “Company”, then we will create a table named “Employee” in the database. After creating a table we will execute the query.

Step 1: Creating a database

```
CREATE DATABASE Company;
```

Step 2: To use this database

```
USE Company;
```

Step 3: Creating a table

```
CREATE TABLE Employee (
  EmployeeId int,
  Name varchar(20),
  Gender varchar(20),
  Salary int,
  Department varchar(20),
  Experience varchar(20)
);
```

Add value to the table:

```
INSERT INTO Employee (EmployeeId, Name, Gender, Salary, Department, Experience)
VALUES (5, 'Priya Sharma', 'Female', 45000, 'IT', '2 years');

INSERT INTO Employee (EmployeeId, Name, Gender, Salary, Department, Experience)
VALUES (6, 'Rahul Patel', 'Male', 65000, 'Sales', '5 years');

INSERT INTO Employee (EmployeeId, Name, Gender, Salary, Department, Experience)
VALUES (7, 'Nisha Gupta', 'Female', 55000, 'Marketing', '4 years');
```

```
INSERT INTO Employee (EmployeeId, Name, Gender, Salary, Department, Experience)
VALUES (8, 'Vikram Singh', 'Male', 75000, 'Finance', '7 years');

INSERT INTO Employee (EmployeeId, Name, Gender, Salary, Department, Experience)
VALUES (9, 'Aarti Desai', 'Female', 50000, 'IT', '3 years');
```

The final table is:

```
SELECT * FROM Employee;
```

Output:

EmployeeId	Name	Gender	Salary	Department	Experience
5	Priya Sharma	Female	45000	IT	2 years
6	Rahul Patel	Male	65000	Sales	5 years
7	Nisha Gupta	Female	55000	Marketing	4 years
8	Vikram Singh	Male	75000	Finance	7 years
9	Aarti Desai	Female	50000	IT	3 years

Step 4: Execute the query

This employee table will help us understand the HAVING Clause. It contains employee IDs, Name, Gender, department, and salary. To Know the sum of salaries, we will write the query:

```
SELECT Department, sum(Salary) as Salary
FROM employee
GROUP BY department;
```

Here is the result,

Output

Department	Salary
Finance	75000
IT	95000
Marketing	55000
Sales	65000

Now if we need to display the departments where the sum of salaries is 50,000 or more. In this condition, we will use the HAVING Clause.

```
SELECT Department, sum(Salary) as Salary
FROM employee
GROUP BY department
HAVING SUM(Salary) >= 50000;
```

Output:

Department	Salary
Finance	75000
IT	95000
Marketing	55000
Sales	65000

Example 2:

Suppose, a teacher wants to announce the toppers in class. For this, she decides to reward every student who scored more than 95%. We need to group the database by name and their percentage and find out who scored more than 95% in that year. So for this first, we create a database name “School”, and then we will create a table named “Student” in the database. After creating a table we will execute the query.

Step 1: Creating a database

```
CREATE DATABASE School;
```

Step 2: To use this database

```
USE School;
```

Step 3: Creating a table

```
CREATE TABLE Student(
   student Varchar(20),
   percentage int
);
```

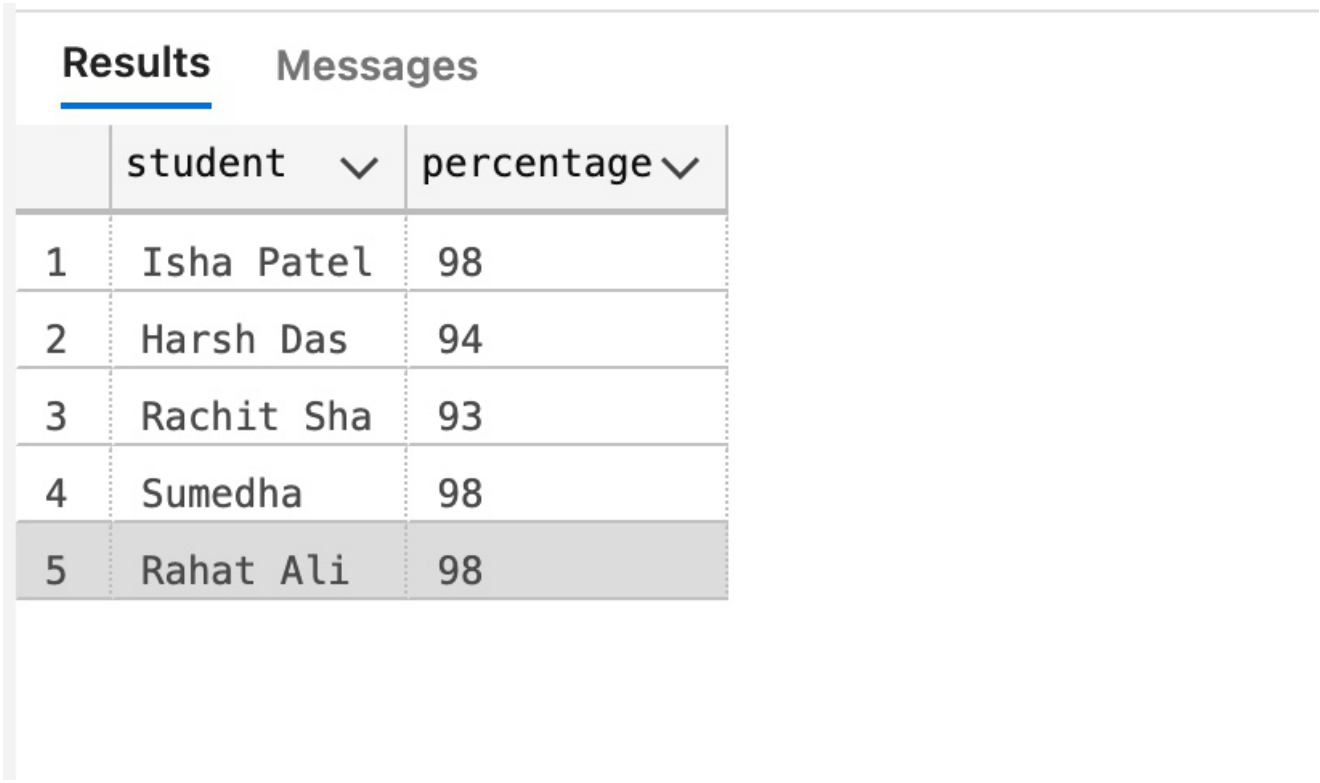
Add value to the table:

```
INSERT INTO Student VALUES ('Isha Patel', 98)
INSERT INTO Student VALUES ('Harsh Das', 94)
INSERT INTO Student VALUES ('Rachit Sha', 93)
INSERT INTO Student VALUES ('Sumedha', 98)
INSERT INTO Student VALUES ('Rahat Ali', 98)
```

The final table is:

```
SELECT * FROM Student;
```

Output:



Results Messages

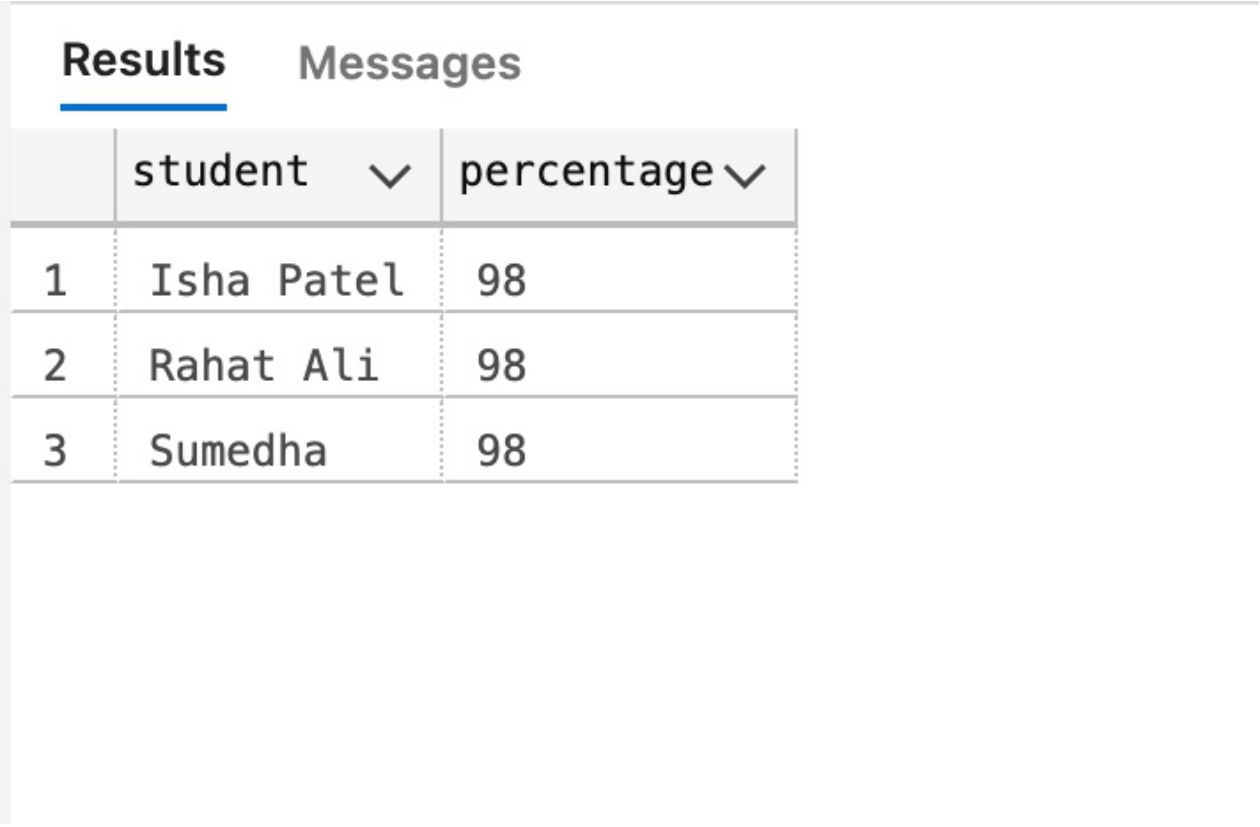
	student	percentage
1	Isha Patel	98
2	Harsh Das	94
3	Rachit Sha	93
4	Sumedha	98
5	Rahat Ali	98

Step 4: Execute Query

```
SELECT student, percentage
FROM Student
GROUP BY student, percentage
HAVING percentage > 95;
```

Here, three students named Isha, Sumedha, and Rahat Ali scored more than 95 %.

Output:


Results Messages

	student	percentage
1	Isha Patel	98
2	Rahat Ali	98
3	Sumedha	98

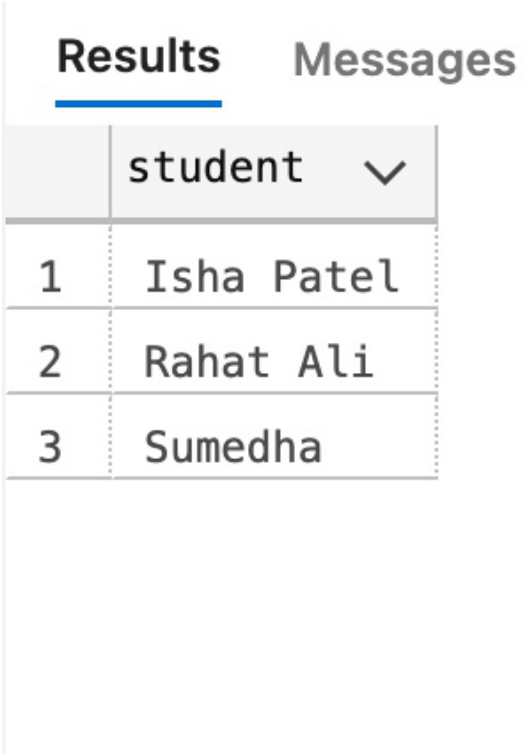
Further, we can also filter rows on multiple values using the HAVING clause. The HAVING clause also permits filtering rows using more than one aggregate condition.

Query:

```
SELECT student
FROM Student
WHERE percentage > 90
GROUP BY student, percentage
HAVING SUM(percentage) < 1000 AND AVG(percentage) > 95;
```

This query returns the students who have more percentage than 95 and the sum of percentage is less than 1000.

Output:


Results Messages

	student
1	Isha Patel
2	Rahat Ali
3	Sumedha

SQL Having vs WHERE

Having	Where
In the HAVING clause it will check the condition in group of a row.	In the WHERE condition it will check or execute at each row individual.
HAVING clause can only be used with aggregate function.	The WHERE Clause cannot be used with aggregate function like Having
Priority Wise HAVING Clause is executed after Group By.	Priority Wise WHERE is executed before Group By.

Last Updated : 18 Apr, 2023

12

Similar Reads

1. Difference between Having clause and Group by clause
2. SQL query using COUNT and HAVING clause
3. Difference between Where and Having Clause in SQL