

## **Assignment-4 Stored procedure**

### **1) MySQL Stored Procedure**

A procedure (often called a stored procedure) is a collection of pre-compiled SQL statements stored inside the database. It is a subroutine or a subprogram in the regular computing language. A procedure always contains a name, parameter lists, and SQL statements.

A procedure is called a recursive stored procedure when it calls itself. Most database systems support recursive stored procedures.

#### **Stored Procedure Features:-**

Stored Procedure increases the performance of the applications. Once stored procedures are created, they are compiled and stored in the database.

Stored procedure reduces the traffic between application and database server. Because the application has to send only the stored procedure's name and parameters instead of sending multiple SQL statements.

Stored procedures are reusable and transparent to any applications.

A procedure is always secure. The database administrator can grant permissions to applications that access stored procedures in the database without giving any permissions on the database tables.

#### **How to create a procedure?**

The following syntax is used for creating a stored procedure in MySQL.

#### **Syntax :-**

DELIMITER &&

```
CREATE PROCEDURE procedure_name [[IN | OUT | INOUT]
parameter_name datatype [, parameter datatype]] ]
```

```
BEGIN
```

```
    Declaration_section
```

Executable\_section

END &&

DELIMITER ;

- procedure\_name => It represents the name of the stored procedure.
- parameter => It represents the number of parameters. It can be one or more than one.
- Declaration\_section => It represents the declarations of all variables.
- Executable\_section => It represents the code for the function execution.

IN parameter:-

It is the default mode. It takes a parameter as input, such as an attribute.

OUT parameters:-

It is used to pass a parameter as output.

INOUT parameters:-

It is a combination of IN and OUT parameters.

### **How to call a stored procedure?**

We can use the CALL statement to call a stored procedure. This statement returns the values to its caller through its parameters (IN, OUT, or INOUT).

The following syntax is used to call the stored procedure in MySQL:

CALL procedure\_name ( parameter(s))

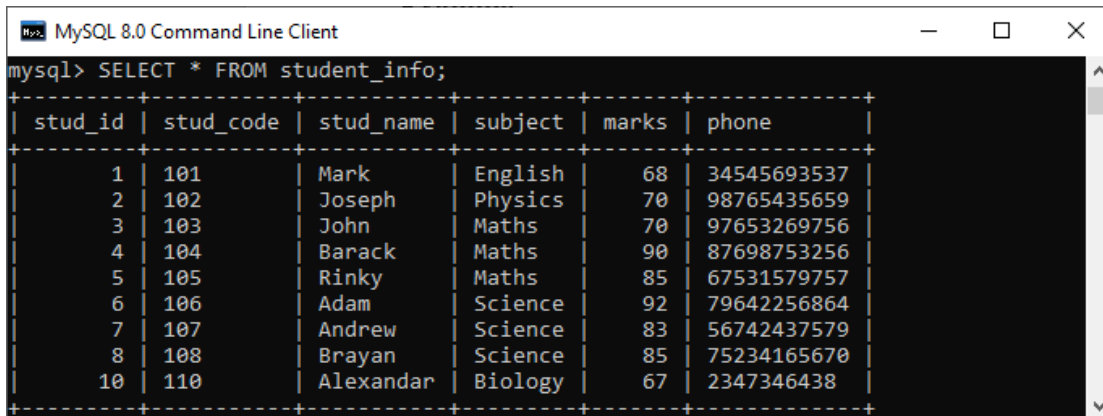
### **Example:-**

Let us understand how to create a procedure in MySQL through example.

First, we need to select a database that will store the newly created procedure. We can select the database using the below statement:

mysql> USE database\_name;

Suppose this database has a table named student\_info that contains the following data:



```
mysql> SELECT * FROM student_info;
```

stud_id	stud_code	stud_name	subject	marks	phone
1	101	Mark	English	68	34545693537
2	102	Joseph	Physics	70	98765435659
3	103	John	Maths	70	97653269756
4	104	Barack	Maths	90	87698753256
5	105	Rinky	Maths	85	67531579757
6	106	Adam	Science	92	79642256864
7	107	Andrew	Science	83	56742437579
8	108	Brayan	Science	85	75234165670
10	110	Alexandar	Biology	67	2347346438

Suppose we want to display all records of this table whose marks are greater than 70 and count all the table rows. The following code creates a procedure named get\_merit\_students:

```
DELIMITER &&
```

```
CREATE PROCEDURE get_merit_student ()
```

```
BEGIN
```

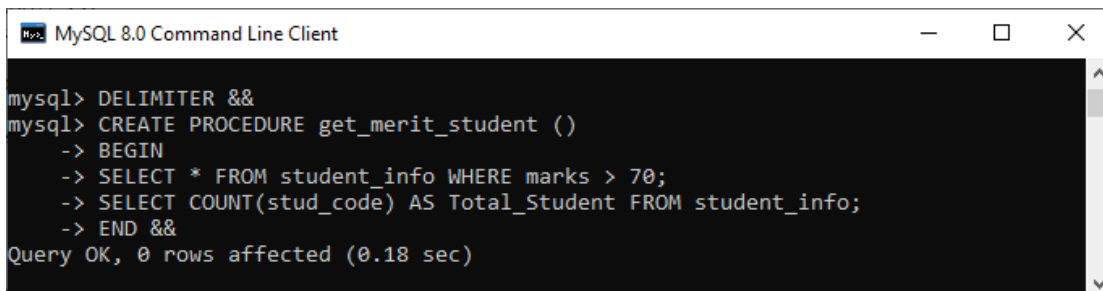
```
    SELECT * FROM student_info WHERE marks > 70;
```

```
    SELECT COUNT(stud_code) AS Total_Student FROM student_info;
```

```
END &&
```

```
DELIMITER ;
```

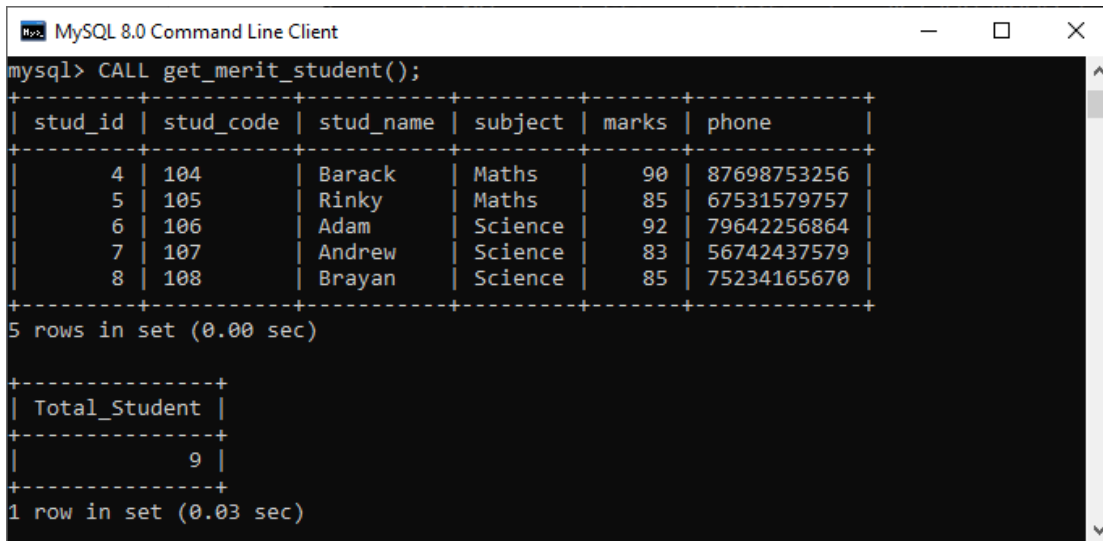
If this code executed successfully, we would get the below output:



```
mysql> DELIMITER &&
mysql> CREATE PROCEDURE get_merit_student ()
-> BEGIN
-> SELECT * FROM student_info WHERE marks > 70;
-> SELECT COUNT(stud_code) AS Total_Student FROM student_info;
-> END &&
Query OK, 0 rows affected (0.18 sec)
```

call the procedure to verify the output:

```
mysql> CALL get_merit_student();
```



```
mysql> CALL get_merit_student();
```

stud_id	stud_code	stud_name	subject	marks	phone
4	104	Barack	Maths	90	87698753256
5	105	Rinky	Maths	85	67531579757
6	106	Adam	Science	92	79642256864
7	107	Andrew	Science	83	56742437579
8	108	Brayan	Science	85	75234165670

5 rows in set (0.00 sec)

Total_Student
9

1 row in set (0.03 sec)

## **2. Explain ACID properties.**

DBMS is the management of data that should remain integrated when any changes are done in it. It is because if the integrity of the data is affected, whole data will get disturbed and corrupted. Therefore, to maintain the integrity of the data, there are four properties described in the database management system, which are known as the ACID properties.

### **ACID Properties:-**

#### **Atomicity, Consistency, Isolation, Durability**

##### **1) Atomicity:-**

The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all. It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.

##### **2) Consistency:-**

The word consistency means that the value should remain preserved always. In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always. In the

case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.

### **3) Isolation:-**

The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently. In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another. In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained. Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.

### **4) Durability:-**

Durability ensures the permanency of something. In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database. The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives.

### **3. Explain offset in limit query with example.**

The limit keyword is used to limit the number of rows returned in a query result.

Example:-

```
SELECT * FROM employees LIMIT 2;
```

The OFF SET value is also most often used together with the LIMIT keyword. The OFF SET value allows us to specify which row to start from retrieving data

Using LIMIT with OFFSET:-

At times we have to mention the offset and no. of rows to be returned by the LIMIT clause.

For example: Starting at offset 5, I want to see the next 3 records.

For such requirements, we can use LIMIT with OFFSET.

```
SELECT * FROM employees LIMIT 5,3;
```

#### **4.Left Join and Right Join with example.**

##### **SQL LEFT JOIN Keyword:-**

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

##### **LEFT JOIN Syntax:-**

```
SELECT column_name(s)
```

```
FROM table1
```

```
LEFT JOIN table2
```

```
ON table1.column_name = table2.column_name;
```

##### **Example:-**

```
SELECT Customers.CustomerName, Orders.OrderID
```

```
FROM Customers
```

```
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
```

```
ORDER BY Customers.CustomerName;
```

##### **SQL RIGHT JOIN Keyword:-**

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

##### **RIGHT JOIN Syntax:-**

```
SELECT column_name(s)
```

FROM table1

RIGHT JOIN table2

ON table1.column\_name = table2.column\_name;

**Example:-**

SELECT Orders.OrderID, Employees.LastName, Employees.FirstName

FROM Orders

RIGHT JOIN Employees ON Orders.EmployeeID =  
Employees.EmployeeID

ORDER BY Orders.OrderID;

**5. Explain Group by**

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

**GROUP BY Syntax:-**

SELECT column\_name(s)

FROM table\_name

WHERE condition

GROUP BY column\_name(s)

ORDER BY column\_name(s);

**Example:-**

SELECT COUNT(CustomerID), Country

FROM Customers

GROUP BY Country  
ORDER BY COUNT(CustomerID) DESC;

## **6. Logical operators in mySQL.**

ALL=> TRUE if all of the subquery values meet the condition

Example:-

```
SELECT * FROM Products  
WHERE Price > ALL (SELECT Price FROM Products WHERE Price >  
500);
```

AND=> TRUE if all the conditions separated by AND is TRUE

Example:-

```
SELECT * FROM Customers  
WHERE City = "London" AND Country = "UK";
```

ANY=> TRUE if any of the subquery values meet the condition

BETWEEN=> TRUE if the operand is within the range of comparison

EXISTS=> TRUE if the subquery returns one or more records

IN=> TRUE if the operand is equal to one of a list of expressions

LIKE=> TRUE if the operand matches a pattern

NOT=> Displays a record if the condition(s) is NOT TRUE

OR=> TRUE if any of the conditions separated by OR is TRUE

SOME=> TRUE if any of the subquery values meet the condition



## 7.Data types in mySQL

### String Data Types:-

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LOBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

### Numeric Data Types:-

Data type	Description
BIT(size)	A bit-value type. The number of bits per value is specified in <i>size</i> . The <i>size</i> parameter can hold a value from 1 to 64. The default value for <i>size</i> is 1.
TINYINT(size)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The <i>size</i> parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(size)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The <i>size</i> parameter specifies the maximum display width (which is 255)
MEDIUMINT(size)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The <i>size</i> parameter specifies the maximum display width (which is 255)
INT(size)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The <i>size</i> parameter specifies the maximum display width (which is 255)
INTEGER(size)	Equal to INT(size)
BIGINT(size)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The <i>size</i> parameter specifies the maximum display width (which is 255)
FLOAT(size, d)	A floating point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(p)	A floating point number. MySQL uses the <i>p</i> value to determine whether to use FLOAT or DOUBLE for the resulting data type. If <i>p</i> is from 0 to 24, the data type becomes FLOAT(). If <i>p</i> is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(size, d)	A normal-size floating point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter
DOUBLE PRECISION(size, d)	
DECIMAL(size, d)	An exact fixed-point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter. The maximum number for <i>size</i> is 65. The maximum number for <i>d</i> is 30. The default value for <i>size</i> is 10. The default value for <i>d</i> is 0.
DEC(size, d)	Equal to DECIMAL(size,d)

## Date and Time Data Types

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME( <i>fsp</i> )	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP( <i>fsp</i> )	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME( <i>fsp</i> )	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

## 8.Float and double, Where to use them.

### Float Datatype:-

It is a 32-bit, single-precision floating-point number. It means that it gives 6-7 decimal digits precision. It is used if we want to use memory effectively because it takes less memory in comparison to double data type. To define a float value, we must use a suffix f or F.

float height = 167.7f

or

float height = 167.7F

### Double Datatype:-

The double data type is a 64-bit double-precision floating-point number. It means that it gives 15-16 decimal digits precision. It consumes more memory in comparison to the float data type. It is used to store decimal values. Its default value is 0.0d. It is optional to add suffix d or D.

For example:

double price = 987.90D

or

double price = 987.90d

or

double price = 987.90

## 9.Functions in MySQL

### MySQL String Functions:-

Function	Description
<a href="#">ASCII</a>	Returns the ASCII value for the specific character
<a href="#">CHAR_LENGTH</a>	Returns the length of a string (in characters)
<a href="#">CHARACTER_LENGTH</a>	Returns the length of a string (in characters)
<a href="#">CONCAT</a>	Adds two or more expressions together
<a href="#">CONCAT_WS</a>	Adds two or more expressions together with a separator
<a href="#">FIELD</a>	Returns the index position of a value in a list of values
<a href="#">FIND_IN_SET</a>	Returns the position of a string within a list of strings
<a href="#">FORMAT</a>	Formats a number to a format like "#,###,###.##", rounded to a specified number of decimal places
<a href="#">INSERT</a>	Inserts a string within a string at the specified position and for a certain number of characters
<a href="#">INSTR</a>	Returns the position of the first occurrence of a string in another string
<a href="#">LCASE</a>	Converts a string to lower-case
<a href="#">LEFT</a>	Extracts a number of characters from a string (starting from left)
<a href="#">LENGTH</a>	Returns the length of a string (in bytes)
<a href="#">LOCATE</a>	Returns the position of the first occurrence of a substring in a string
<a href="#">LOWER</a>	Converts a string to lower-case
<a href="#">LPAD</a>	Left-pads a string with another string, to a certain length
<a href="#">LTRIM</a>	Removes leading spaces from a string
<a href="#">MID</a>	Extracts a substring from a string (starting at any position)
<a href="#">POSITION</a>	Returns the position of the first occurrence of a substring in a string
<a href="#">REPEAT</a>	Repeats a string as many times as specified
<a href="#">REPLACE</a>	Replaces all occurrences of a substring within a string, with a new substring
<a href="#">REVERSE</a>	Reverses a string and returns the result

### MySQL Numeric Functions:-

Function	Description
<a href="#"><u>ABS</u></a>	Returns the absolute value of a number
<a href="#"><u>ACOS</u></a>	Returns the arc cosine of a number
<a href="#"><u>ASIN</u></a>	Returns the arc sine of a number
<a href="#"><u>ATAN</u></a>	Returns the arc tangent of one or two numbers
<a href="#"><u>ATAN2</u></a>	Returns the arc tangent of two numbers
<a href="#"><u>AVG</u></a>	Returns the average value of an expression
<a href="#"><u>CEIL</u></a>	Returns the smallest integer value that is >= to a number
<a href="#"><u>CEILING</u></a>	Returns the smallest integer value that is >= to a number
<a href="#"><u>COS</u></a>	Returns the cosine of a number
<a href="#"><u>COT</u></a>	Returns the cotangent of a number
<a href="#"><u>COUNT</u></a>	Returns the number of records returned by a select query
<a href="#"><u>DEGREES</u></a>	Converts a value in radians to degrees
<a href="#"><u>DIV</u></a>	Used for integer division
<a href="#"><u>EXP</u></a>	Returns e raised to the power of a specified number
<a href="#"><u>FLOOR</u></a>	Returns the largest integer value that is <= to a number
<a href="#"><u>GREATEST</u></a>	Returns the greatest value of the list of arguments
<a href="#"><u>LEAST</u></a>	Returns the smallest value of the list of arguments
<a href="#"><u>LN</u></a>	Returns the natural logarithm of a number
<a href="#"><u>LOG</u></a>	Returns the natural logarithm of a number, or the logarithm of a number to a specified base
<a href="#"><u>LOG10</u></a>	Returns the natural logarithm of a number to base 10
<a href="#"><u>LOG2</u></a>	Returns the natural logarithm of a number to base 2
<a href="#"><u>MAX</u></a>	Returns the maximum value in a set of values

## MySQL Date Functions

Function	Description
<a href="#"><u>ADDDATE</u></a>	Adds a time/date interval to a date and then returns the date
<a href="#"><u>ADDTIME</u></a>	Adds a time interval to a time/datetime and then returns the time/datetime
<a href="#"><u>CURDATE</u></a>	Returns the current date
<a href="#"><u>CURRENT_DATE</u></a>	Returns the current date
<a href="#"><u>CURRENT_TIME</u></a>	Returns the current time
<a href="#"><u>CURRENT_TIMESTAMP</u></a>	Returns the current date and time
<a href="#"><u>CURTIME</u></a>	Returns the current time
<a href="#"><u>DATE</u></a>	Extracts the date part from a datetime expression
<a href="#"><u>DATEDIFF</u></a>	Returns the number of days between two date values
<a href="#"><u>DATE_ADD</u></a>	Adds a time/date interval to a date and then returns the date
<a href="#"><u>DATE_FORMAT</u></a>	Formats a date
<a href="#"><u>DATE_SUB</u></a>	Subtracts a time/date interval from a date and then returns the date
<a href="#"><u>DAY</u></a>	Returns the day of the month for a given date
<a href="#"><u>DAYNAME</u></a>	Returns the weekday name for a given date
<a href="#"><u>DAYOFMONTH</u></a>	Returns the day of the month for a given date
<a href="#"><u>DAYOFWEEK</u></a>	Returns the weekday index for a given date
<a href="#"><u>DAYOFYEAR</u></a>	Returns the day of the year for a given date
<a href="#"><u>EXTRACT</u></a>	Extracts a part from a given date
<a href="#"><u>FROM_DAYS</u></a>	Returns a date from a numeric datevalue
<a href="#"><u> HOUR</u></a>	Returns the hour part for a given date
<a href="#"><u>LAST_DAY</u></a>	Extracts the last day of the month for a given date
<a href="#"><u>LOCALTIME</u></a>	Returns the current date and time

## **10. Having clause.**

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

### **HAVING Syntax:-**

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

### **Example:-**

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

