

Structured Query Language (SQL)

(Chapter 11)

SORTING IN SQL—ORDER BY

The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. The ORDER BY keyword is used to sort the result-set by one or more fields in a table. This clause sorts the records in the ascending order (ASC) by default. Therefore, in order to sort the records in descending order, DESC keyword is to be used. Sorting using ORDER BY clause can be done on multiple columns, separated by comma.

Syntax for ORDER BY clause:

```
SELECT <column-list> FROM <table_name> [WHERE <condition>]  
ORDER BY <column_name> [ASC|DESC];
```

Here, WHERE clause is optional.

For example,

- To display the roll number, name and marks of students on the basis of their marks in the ascending order.
mysql> SELECT Rollno, Name, Marks FROM student ORDER BY Name;

- To display the roll number, name and marks of all the students in the descending order of their marks and ascending order of their names.

```
mysql> SELECT Rollno, Name, Marks FROM student  
ORDER BY Marks DESC, Name;
```

Rollno	Name	Marks
8.	Akshay Dureja	90
3.	Ankit Sharma	76
2.	Deep Singh	98
6.	Diksha Sharma	80
7.	Gurpreet Kaur	NULL
5.	Payal Goel	82
10.	Prateek Mittal	75
4.	Radhika Gupta	78
1.	Raj Kumar	93
9.	Shreya Anand	70

Sorting on Column Alias

If a column alias is defined for a column, it can be used for displaying rows in ascending or descending order using ORDER BY clause.

For example, SELECT Rollno, Name, Marks AS Marks_obtained
FROM student
ORDER BY Marks_obtained;

Alias name

AGGREGATE FUNCTIONS

Till now we have studied about single row functions which work on a single value. SQL also provides multiple row functions which work on multiple values. So, we can apply SELECT query on a group of records rather than the entire table. Therefore, these functions are called aggregate functions or group functions.

Generally, the following aggregate functions are applied on groups as described below:

Table 1: Aggregate Functions in SQL

S.No.	Function	Description/Purpose
1.	MAX()	Returns the maximum/highest value among the values in the given column/expression.
2.	MIN()	Returns the minimum/lowest value among the values in the given column/expression.
3.	SUM()	Returns the sum of the values under the specified column/expression.
4.	AVG()	Returns the average of the values under the specified column/expression.
5.	COUNT()	Returns the total number of values/records under the specified column/expression.



Consider a table Employee (Employee code, employee name, job, salary and city) with the following structure:

Ecode	Name	Salary	Job	City
E1	Ritu Jain	5000	Manager	Delhi
E2	Vikas Verma	4500	Executive	Jaipur
E3	Rajat Chaudhary	6000	Clerk	Kanpur
E4	Leena Arora	7200	Manager	Bangalore
E5	Shikha Sharma	8000	Accountant	Kanpur

1. MAX()

MAX() function is used to find the highest value among the given set of values of any column or expression based on column. MAX() takes one argument which can be either a column name or any valid expression involving a particular column from the table.

For example,

```
mysql> Select MAX(Salary) from EMPLOYEE;
```

Output:

```
+-----+
| MAX(Salary) |
+-----+
| 8000        |
+-----+
```

This command on execution shall return the maximum value from the specified column (Salary) of the table Employee, which is 8000.

2. MIN()

MIN() function is used to find the lowest value among the given set of values of any column or expression based on column. MIN() takes one argument which can be either a column name or any valid expression involving a particular column from the table.

For example,

```
mysql> Select MIN(Salary) from EMPLOYEE;
```

Output:

```
+-----+
| MIN(Salary) |
+-----+
| 4500        |
+-----+
```

This command on execution shall return the minimum value from the specified column (Salary) of the table Employee, which is 4500.



3. SUM()

SUM() function is used to find the total value of any column or expression based on a column. It accepts the entire range of values as an argument, which is to be summed up on the basis of a particular column, or an expression containing that column name. The SUM() function always takes argument of integer type only. Sums of String and Date type data are not defined.

For example,

```
mysql> Select SUM(Salary) from EMPLOYEE;
```

```
+-----+
| SUM(Salary) |
+-----+
| 30700       |
+-----+
```

This command on execution shall return the total of the salaries of all the employees from the specified column (Salary) of the table Employee, which is 30700.

4. AVG()

AVG() function is used to find the average value of any column or expression based on a column. Like sum(), it also accepts the entire range of values of a particular column to be taken average of, or even a valid expression based on this column name. Like SUM() function, the AVG() function always takes argument of integer type only. Average of String and Date type data is not defined.

For example,

```
mysql> Select AVG(Salary) from EMPLOYEE;
```

```
+-----+
| AVG(Salary) |
+-----+
| 6140        |
+-----+
```

This command on execution shall return the average of the salaries of all the employees from the specified column (Salary) of the table Employee, which is 6140.

5. COUNT()

COUNT() function is used to count the number of values in a column. COUNT() takes one argument which can be any column name, or an expression based on a column, or an asterisk (*). When the argument is a column name or an expression based on column, COUNT() returns the number of non-NULL values in that column. If the argument is asterisk (*), then COUNT() counts the total number of records/rows satisfying the condition, if any, in the table.

For example,

```
1. mysql> Select COUNT(*) from EMPLOYEE;
```

```
+-----+
| COUNT(*)  |
+-----+
| 5         |
+-----+
```

This command on execution shall return the total number of records in the table Employee, which is 5.

2. **mysql> Select COUNT(DISTINCT City) from EMPLOYEE;**

```
+-----+
| COUNT(DISTINCT City) |
+-----+
| 4                     |
+-----+
```

This command on execution shall return the total number of records on the basis of city with no duplicate values, *i.e.*, Kanpur is counted only once; the second occurrence is ignored by MySQL because of the DISTINCT clause. Thus, the output will be 4 instead of 5.



➤ **Aggregate Functions & NULL Values**

Consider the following table Employee with NULL values against the Salary field for some employees:

Employee

Ecode	Name	Salary	Job	City
E1	Ritu Jain	NULL	Manager	Delhi
E2	Vikas Verma	4500	Executive	Jaipur
E3	Rajat Chaudhary	6000	Clerk	Kanpur
E4	Leena Arora	NULL	Manager	Bangalore
E5	Shikha Sharma	8000	Accountant	Kanpur

None of the aggregate functions takes NULL into consideration. NULL values are simply ignored by all the aggregate functions as clearly shown in the examples given below:

mysql> Select Sum(Salary) from Employee;

Output: 18500

mysql> Select Min(Salary) from Employee;

Output: 4500 (NULL values are not considered.)

mysql> Select Max(Salary) from Employee;

Output: 8000 (NULL values are not ignored.)

mysql> Select Count(Salary) from Employee;

Output: 3 (NULL values are not ignored.)

mysql> Select Avg(Salary) from Employee;

Output: 6166.66 (It will be calculated as 18500/3, *i.e.*, sum/total no. of records, which are 3 after ignoring NULL values.)

mysql> Select Count(*) from Employee;

Output: 5

mysql> Select Count(Ecode) from Employee;

Output: 5 (No NULL value exists in the column Ecode.)

mysql> Select Count(Salary) from Employee;

Output: 3 (NULL values are ignored while counting the total number of records on the basis of Salary.)

SOLVED QUESTIONS

1. Consider the following two tables: STATIONERY and CONSUMER.

Table: Stationery

S_ID	StationeryName	Company	Price	StockDate
DP01	Dot Pen	ABC	10	2011-03-31
PL02	Pencil	XYZ	6	2010-01-01
ER05	Eraser	XYZ	7	2010-02-14
PL01	Pencil	CAM	5	2009-01-09
GP02	Gel Pen	ABC	15	2009-03-19

Note:

- S_ID is the Primary Key.

Table: Consumer

C_ID	ConsumerName	Address	P_ID
01	Good Learner	Delhi	PL01
06	Write Well	Mumbai	GP02
12	Topper	Delhi	DP01
15	Write & Draw	Delhi	PL02
16	Motivation	Bengaluru	PL01

Note:

- C_ID is the Primary Key.
- P_ID is the Foreign Key referencing S_ID of Stationery table.

Write SQL statements for the queries (i) to (iii) and output for (iv) and (v):

- To display details of all the Stationery items in the Stationery table in descending order of StockDate.
- To display details of that Stationery item whose Company is XYZ and price is below 10.
- To increase the price of all the Stationery items in Stationery table by 2.
- Select COUNT(DISTINCT Address) from Consumer;
- Select StationeryName, price * 3 from Stationery where Company = 'CAM';

- Ans.**
- Select * from Stationery order by StockDate desc;
 - Select * from Stationery where Company = 'XYZ' and Price < 10;
 - Update Stationery Set Price = Price + 2;
 - 3
 - Pencil 15

2. Answer the questions given below on the the basis of the following tables SHOP and ACCESSORIES.

Table: SHOP

Id	SName	Area
S01	ABC Computronics	CP
S02	All Infotech Media	GK II
S03	Tech Shoppe	CP
S04	Geek Tenco Soft	Nehru Place
S05	Hitech Tech Store	Nehru Place

Table: ACCESSORIES

No	Name	Price	Id
A01	Motherboard	12000	S01
A02	Hard Disk	5000	S01
A03	Keyboard	500	S02
A04	Mouse	300	S01
A05	Motherboard	13000	S02
A06	Keyboard	400	S03
A07	LCD	6000	S04
T08	LCD	5500	S05
T09	Mouse	350	S05
T010	Hard Disk	450	S03



- (a) Write the SQL queries:
- (i) To display Name and Price of all the Accessories in ascending order of their Price.
 - (ii) To display Id and SName of all Shops located in Nehru Place.
 - (iii) To display Minimum and Maximum Price of all the accessories.

Ans. (i) SELECT Name, Price FROM ACCESSORIES ORDER BY Price;
(ii) SELECT Id, SName FROM SHOP WHERE Area='Nehru Place';
(iii) SELECT Name, MAX(Price), MIN(Price) FROM ACCESSORIES;
(b) Write the output of the following SQL commands:
(i) SELECT DISTINCT NAME FROM ACCESSORIES WHERE PRICE>=5000;
(ii) SELECT AREA, COUNT(*) FROM SHOP GROUP BY AREA;
(iii) SELECT COUNT(DISTINCT AREA) FROM SHOP;
(iv) SELECT NAME, PRICE*0.05 DISCOUNT FROM ACCESSORIES WHERE SNO IN ('S02', 'S03');

Ans. (i) Name
Motherboard
Hard Disk
LCD
(ii) AREA COUNT
CP 2
GK II 1
Nehru Place 2
(iii) COUNT 3
(iv) NAME PRICE
Keyboard 25.00
Motherboard 650.00
Keyboard 20.00
Hard Disk 225.00

3. Consider the following tables CARDEN and CUSTOMER and answer the question:

Table: CARDEN

Ccode	CarName	Make	Colour	Capacity	Charges
501	A-Star	Suzuki	RED	3	14
503	Indigo	Tata	SILVER	3	12
502	Innova	Toyota	WHITE	7	15
509	SX4	Suzuki	SILVER	4	14
510	C Class	Mercedes	RED	4	35

Table: CUSTOMER

Ccode	CName	Ccode
1001	Hemant Sahu	501
1002	Raj Lal	509
1002	Feroza Shah	503
1004	Ketan Dhal	502

- (a) Write SQL commands for the following statements:
- (i) To display names of all the silver-coloured cars.
 - (ii) To display the name, make and capacity of cars in descending order of their sitting capacity.
 - (iii) To display the highest charges at which a vehicle can be hired from CARDEN.

Ans. (i) SELECT CarName FROM carden WHERE Colour LIKE 'Silver';
(ii) SELECT CarName, Make, Capacity FROM carden ORDER BY Capacity DESC;
(iii) SELECT MAX(Charges) FROM carden;
(b) Give the output of the following SQL queries:
(i) SELECT COUNT(DISTINCT Make) FROM CARDEN;
(ii) SELECT MAX(Charges), MIN(Charges) FROM CARDEN;
(iii) SELECT COUNT(*), Make FROM CARDEN;
(iv) SELECT CarName FROM CARDEN WHERE Capacity=4;

Ans. (i) COUNT(DISTINCT Make)
4
(ii) MAX(Charges) MIN(Charges)
35 12

- (iii) COUNT(*) Make
5 Suzuki
- (iv) CarName
SX4
C Class

UNSOLVED QUESTIONS

1. Consider the following tables STORE and SUPPLIERS. Write SQL commands for the statements (a) to (d) and give outputs for SQL queries (e) to (g).

Table: STORE

ItemNo	Item	Scode	Qty	Rate	LastBuy
2005	Sharpener Classic	23	60	8	31-Jun-09
2003	Ball Pen 0.25	22	50	25	01-Feb-10
2002	Gel Pen Premium	21	150	12	24-Feb-10
2006	Gel Pen Classic	21	250	20	11-Mar-09
2001	Eraser Small	22	220	6	19-Jan-09
2004	Eraser Big	22	110	8	02-Dec-09
2009	Ball Pen 0.5	21	180	18	03-Nov-09

Table: SUPPLIERS

Scode	Sname
21	Premium Stationery
23	Soft Plastics
22	Tetra Supply

- (a) To display details of all the items in the Store table in ascending order of LastBuy.
 (b) To display Itemno and item name of those items from store table whose rate is more than 15 rupees.
 (c) To display the details of those items whose supplier code is 22 or Quantity in store is more than 110 from the table Store.
 (d) To display minimum rate of items for each Supplier individually as per Scode from the table Store.
 (e) SELECT COUNT(DISTINCT Scode) FROM STORE;
 (f) SELECT Rate*Qty FROM STORE WHERE Itemno=2004;
 (g) SELECT MAX>LastBuy)FROM STORE;

2. Consider the given table and answer the questions.

Table: SchoolBus

Rtno	Area_Covered	Capacity	Noofstudents	Distance	Transporter	Charges
1	Vasant Kunj	100	120	10	Shivam travels	100000
2	Hauz Khas	80	80	10	Anand travels	85000
3	Pitampura	60	55	30	Anand travels	60000
4	Rohini	100	90	35	Anand travels	100000
5	Yamuna Vihar	50	60	20	Bhalla Co.	55000
6	Krishna Nagar	70	80	30	Yadav Co.	80000
7	Vasundhara	100	110	20	Yadav Co.	100000
8	Paschim Vihar	40	40	20	Speed travels	55000
9	Saket	120	120	10	Speed travels	100000
10	Janakpuri	100	100	20	Kisan Tours	95000

- (a) To show all information of students where capacity is more than the no. of students in order of Rtno.
 (b) To show Area_covered for buses covering more than 20 km., but charges less than 80000.
 (c) To show transporter-wise total no. of students travelling.
 (d) To show Rtno, Area_covered and average cost per student for all routes where average cost per student is—Charges/Noofstudents.
 (e) Add a new record with the following data:
 (11, "Motibagh",35,32,10, "kisan tours", 35000)
 (f) Give the output considering the original relation as given:
 (i) select sum(distance) from schoolbus where transporter= "Yadav travels";
 (ii) select min(noofstudents) from schoolbus;
 (iii) select avg(charges) from schoolbus where transporter= "Anand travels";
 (g) Select distinct transporter from schoolbus;